Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Maja Zadnikar

Sistem za optimizacijo sledilnikov v računalniškem vidu

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Matej Kristan

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Vizualno sledenje objektov je trenutno izredno aktivno področje računalniškega vida, o čemer pričajo številni pregledni članki ter članki z izvirnimi sledilniki. Letno se zgolj na glavnih konferencah računalniškega vida objavi preko dvajset novih sledilnih algoritmov. Večina teh algoritmov vsebuje številne parametre, katere avtorji nastavijo ročno. Ne obstaja pa formalni postopek analize vpliva teh parametrov na uspešnost sledenja, kakor tudi ne sistem za tako analizo. V diplomski nalogi obravnavajte problem avtomatske analize vpliva parametrov sledilnih algoritmov. Predlagajte ogrodje za tako analizo kakor tudi metode za preiskovanje prostora parametrov ter vizualizacijo njihovega vpliva. Postopek demonstrirajte na izbranem sledilniku.

Zahvala gre mentorju doc. dr. Mateju Kristanu, za vso pomoč, podporo in potrpežjivost. Zahvaljujem se as. dr. Luki Čehovinu, za tehnično pomoč in napotke.
Na tem mestu bi se rada zahvalila tudi vsem tistim, ki ste me tekom študija vzpodbujali in verjeli vame.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
MS	Mean Shift	Algoritem mean shift
ASMS	Adaptive Scale Mean Shift	Algoritem mean shift z upoštevanjem velikosti
PDF	Probability density function	Funkcija porazdelitve verjetnosti gostote
KDE	Kernel Density Estimate	Ocena gostote jedra
HC	Hill Climbing	Vzpon po hribu
AHC	Any-ascent Hill-Climbing	Vzpon v katerikoli smeri
SHC	Stochastic Hill-Climbing	Stohastični
FAHC	First-ascent Hill-Climbing	Prvi vzpon
SAHC	Steepest-ascent Hill-Climbing	Najstrmejši vzpon

Povzetek

Naslov: Sistem za optimizacijo sledilnikov v računalniškem vidu

Večina sledilnih algoritmov vsebuje številne parametre, s katerimi lahko občutno vplivamo na uspešnost sledenja. Avtorji sledilnikov te največkrat nastavljajo sami, saj formalnega postopka za določitev optimalnih vrednosti parametrov ni. V diplomski nalogi se osredotočimo na razvoj sistema za sistematično analizo parametrov sledilnih algoritmov. Predlagamo iterativni pristop k reševanju naloge optimizacije. S pomočjo preiskovalne metode najstrmejšega vzpona po hribu, v vsaki iteraciji preiskovalni prostor zmanjšamo in se premaknemo v smer najboljše rešitve. Optimizacija temelji na nedavni meri za oceno uspešnosti sledilnikov, ki upošteva reinicializacijo sledilnika in predstavlja pričakovano povprečno prekrivanje regije. Predlagane metode implementiramo z razširitvijo obstoječega sistema za evaluacijo sledilnikov. Predlagamo postopke za vizualizacijo rezultatov, ki nam omogočajo podrobnejšo analizo parametrov in njihovih lastnosti. Predlagamo tudi postopek za ocenjevanje uspešnosti izbrane metode. Razviti sistem in metode nato preizkusimo na primeru analize sledilnega algoritma Mean Shift. Primerjava rezultatov pridobljenih z uporabo predlaganega optimizacijskega pristopa je pokazala, da je uspešnost sledenja možno izboljšati z dobro nastavitvijo parametrov. V primerjavi s parametri, nastavljenimi s strani avtorja, so eksperimentalni rezultati pokazali povprečno izboljšanje ocene sledenja za 5%.

Ključne besede: računalniški vid, sledenje, optimizacija, analiza parametrov.

Abstract

Title: A system for computer-vision-based tracking evaluation

A majority of tracking algorithms have a number of parameters that require tuning and often significantly influence tracking performance. Without a formal procedure of finding the best parameter values, the authors usually set them manually by trial and erorr. In this thesis we focus on development of a system for systematical analysis of trackers parameters. We propose an iterative approach to solving the parameter optimization task. With steepest ascent hill climbing search method, we narrow the search space in each iteration and move in a direction of the best solution. The optimization is based on a recent tracker evaluation measure which considers tracker reinitialization and represents the expected average overlap. We implement the proposed methods by expanding an existing tracking evaluation system. We propose procedures for result visualization, which enable a detailed analysis and insight into parameter properties. We also propose a procedure for determining how successful the optimization method is. As a proof of concept, we test our system and proposed methods on a Mean Shift tracker. By comparing results obtained by our optimization approach, we conclude that we can improve tracker performance by improving the parameter values. Experimental results indicate approximately 5% improvements over original published parameters.

Keywords: computer vision, tracking, optimization, parameters analysis.

Poglavje 1

Uvod in motivacija

Vizualno sledenje objektov je zaradi širokega spektra uporabe na različnih področjih zanimivo, saj ponuja ogromno novih izzivov. Področja, kjer je uporaba vizualnega sledenja še posebej očitna so video nadzorni sistemi, mobilna robotika, navigacija in še veliko več. Sledenje lahko opišemo kot računanje tirnice premikajočih se objektov, skozi sekvenco zaporednih slik. Objekt, ki ga želimo slediti, predstavimo z izbranim modelom in nato v naslednjih slikah iščemo regijo, ki se z modelom najbolj sklada. Težave pri sledenju največkrat povzročijo spremembe objekta ali okolice. Objekt lahko spremeni obliko, velikost, barvo ali ga prekriva drug objekt. Pojavijo se lahko tudi spremembe v svetlobi ali premiku kamere, kar vpliva na zahtevnost sledenja in posledično, učinkovitost izbranega sledilnega algoritma. V splošnem, sledenje začnemo z inicializacijo referenčnega modela, nato v vsaki naslednji sliki določimo kandidate in izračunamo novo lokacijo objekta, kjer je bila podobnost med kandidatom in modelom največja. Na uspešnost sledenja je možno vplivati tudi s skrbnim izborom vrednosti parametrov sledilnika [13], ki predstavlja nov problem. Ker ustaljenega postopka oziroma metode določanja optimalnega nabora vrednosti parametrov ni in je glede na sledilnik lahko teh parametrov veliko, je iskanje po večini zelo zahtevno. Zaradi zahtevnosti preiskovanja prostora vrednosti parametrov so zato potrebni postopki, ki bodo samodejno določili optimalni nabor. V nalogi se osredotočimo predvsem na iskanje ta-

kega postopka in analizo rezultatov.

1.1 Sorodna dela

Na temo sledenja je bilo objavljenih že ogromno del [8,27,28,31,37,43], vendar jih le malo posveti vso pozornost v analizo parametrov in določanja njihovih vrednosti. Izbor optimalnega nabora vrednosti parametrov sledilnika ni enostaven, še posebej če ne poznamo karakteristik objekta in okolice v sekvenci, na kateri se bo sledenje izvajalo. S pravilno nastavljenimi parametri, lahko v primerjavi z nekaterimi najpomembnejšimi (ang., state-of-the-art) sledilniki, presežemo uspešnost sledenja na isti sekvenci [14]. Predstavljena je bila metoda samodejnega iskanja parametrov glede na značilke trenutne slike. V želji po hitrem določanju vrednosti parametrov, metoda predhodno ustvari učno množico parametrov za različne vrednosti značilk, ki določajo sekvenco [13]. Za ocenjevanje sledilnikov obstaja veliko metod, vendar enotnega postopka, ki bi veljal za vse sledilnike in sekvence ni [8,26,41]. Metode pokrivajo več različnih aspektov ocenjevanja, kot sta na primer robustnost in natančnost [8]. Robustnost temelji predvsem na tem, kako dobro sledilnik prenaša spremembe v okolici in na objektu, ki ga sledimo. Pri slabi robustnosti bo sledilnik najverjetneje odpovedal že pri razmeroma majhnih spremembah. Z uporabo le mere robustnosti, pa izpustimo pomemben del ocenjevanja uspešnosti in sicer, kako natančno sledilnik določi novo lokacijo objekta. Za zagotavljanje kar se da realnega vrednotenja je priporočena uporaba več mer za ocenjevanje. Nekaj najbolj uveljavljenih metod za določanje natančnosti predstavljajo presek regije (ang., overlap) [31], napaka centralne točke (ang., center error) [37], natančnost, ki temelji na slikovnih elementih (ang., pixel-based precision) ter ocena F (ang., F-score) [21]. Za merjenje robustnosti pa se velikokrat uporablja mera števila neuspehov (ang., failure rate) [22,27], ter dolžina uspešnega sledenja (ang,. tracking length) [28]. Ker je primerjanje sledilnikov z več različnimi merami težko, se uporabljajo tudi hibridne mere, ki združujejo več mer in nam vrnejo en rezultat. Ena izmed

teh je kombinirana mera za ocenjevanje sledilnikov (ang., Combined Tracking Performance Score, CoTPS) [8, 32, 33]. Za realnejšo oceno je testiranje sledilnika na sekvencah različnih težavnosti in karakteristik ključnega pomena. Zaželjena je uporaba javno dostopnih zbirk kot je VOT (ang., Visual Object Tracking) [35], OTB (ang., Object Tracking Benchmark) [42] in NUS-PRO (ang., NUS - People and Rigid Objects dataset) [29, 30]. To omogoča lažjo primerjavo med sledilniki in preverljivost rezultatov. Čeprav obstaja kar nekaj sistemov za evaluacijo sledilnikov, kot na primer VOT toolkit (Visual Object Tracking toolkit) [6, 12] in VTB (Visual Tracker Benchmark) [42], sistema za samodejno analizo parametrov sledilnika ni. Ta problem je delno naslovljen v sistemu TraXtor [7], ki omogoča primerjavo uspešnosti sledilnika glede na vrednosti izbranega parametra. TraXtor je sistem v razvoju in v primerjavi z ostalimi sistemi podpira paralelno procesiranje.

1.2 Prispevki

Glavni prispevek diplomske naloge je sistem za analizo sledilnih algoritmov. Predlagana je optimizacijska metoda za iskanje optimalnega nabora vrednosti parametrov, pri katerem je sledenje najbolj uspešno. Predlagamo novo hibridno metodo ocenjevanja sledilnikov, ki kot prva, upošteva reinicializacijo sledilnika ob odpovedi. Dodatni prispevek diplomske naloge je integracija izbrane optimizacijske metode in ocenjevanja sledilnika v obstoječi sistem TraXtor [7]. Razširitev je praktično uporabna in omogoča iskanje optimalnih vrednosti parametrov za poljuben sledilnik. Predlagane so tudi različne metode za vizualizacijo rezultatov in analizo parametrov, ki nam omogočajo širši pregled nad pomembnostjo njihovega skrbnega izbora.

1.3 Struktura naloge

Diplomska naloga je razdeljena na pet poglavij. V Poglavju 2 so opisana uporabljena orodja in teoretično ozadje uporabljenih metod. V Poglavju 3 je predstavljen naš pristop k reševanju problema in nadgradnja sistema Traxtor, z novo funkcionalnostjo zagona sledilnikov. Poglavje 4 opisuje način testiranja integriranega sledilnika in predstavi dobljene rezultate. Na koncu, v Poglavju 5, predstavimo še sklepne ugotovitve in možnosti izboljšave ter nadaljne delo.

Poglavje 2

Metode

Poglavje je razdeljeno na pet podpoglavij, ki predstavijo uporabljena orodja in teoretično ozadje metod. V Podpoglavju2.1 je predstavljen evaluacijski sistem TraXtor in način njegove uporabe. Protokol za komunikacijo med sistemom in sledilnikom podrobneje opišemo v Podpoglavju 2.2 in nato, v Podpoglavju 2.3 predstavimo mere ocenjevanja sledilnika. Teoretično ozadje sledilnika uporabljenega v sklopu diplomske naloge, je opisano v Podpoglavju 2.4. Na koncu, v Podpoglavju 2.5 se osredotočimo na nalogo optimizacije in pristopa k reševanju problema.

2.1 Evaluacijski sistem TraXtor

TraXtor [7] je sistem za evaluacijo sledilnikov, ki so ga razvili v Laboratoriju za umetne vizualne spoznavne sisteme na Fakulteti za računalništvo in informatiko. Implementiran je v objektno orientiranem programskem jeziku Java, kar omogoča dobro prenosljivost in enako delovanje na večini operacijskih sistemov. Odvisen je od knjižnice CoffeeShop [4] in anotatorja sekvenc Aibu [3, 7]. CoffeeShop je prostostoječa knjižnica in je skupek javanskih razredov za splošno uporabo, ki omogočajo enostavnejši razvoj manjših ali srednje velikih javanskih aplikacij. Aibu je odvisen od knjižnice CoffeeShop in omogoča branje anotacij ter prikaz rezultatov.

Sistem TraXtor je v pomoč pri razvoju sledilnikov, saj nudi enostaven in hiter pregled nad uspešnostjo sledenja. V primerjavi z že uveljavljenim orodjem za evaluacijo sledilnikov, orodjem VOT Toolkit [3], TraXtor podpira paralelno procesiranje individualnih opravil in testiranje sledilnika z različnimi nabori vrednosti parametrov, kar vodi v hitrejše testiranje. Za lažjo integracijo sledilnikov v sistem, se za kominikacijo med sledilnikom in aplikacijo uporablja protokol TraX [45].

Sistem omogoča dodajanje sekvenc ter sledilnikov in v osnovi ponuja štiri vrste eksperimentov, s katermi lahko izvedemo evaluacijo sledilnika:

1. Primerjalni (ang., Comparative),
2. interaktivni (ang., Interactive sandbox),
3. preiskovalni (ang., Parameter sweep) in
4. dvojno preiskovalni (ang., Dual sweep).

Primerjalni eksperiment ponuja možnost testiranja izbranih sledilnikov na večjem številu sekvenc in omogoča primerjavo pridobljenih rezultatov uspešnosti sledenja na posamezni sekvenci. Interaktivni eksperiment je namenjen predvsem pregledu poteka sledenja izbranega sledilnika, na posamezni sekvenci in omogoča njegov grafični prikaz za vsako zaporedno sliko posebej. Za testiranje uspešnosti sledenja pri različnih vrednostih parametra, je primeren preiskovalni eksperiment, ki omogoča testiranje enega parametra in dvojno preiskovalni ekspriment, ki omogoča testiranje sledilnika na vrednostih dveh parametrov. V vseh zgoraj naštetih eksperimentih sta glavni meri uspešnosti, število odpovedi sledilnika ter povprečni presek regije (ang., average overlap) [8].

2.1.1 Uporaba sistema TraXtor

Ob zagonu aplikacije se odpre okno, kjer lahko izberemo nedavni projekt, kreiramo novega ali naložimo že obstoječega. Za kreiranje in shranjevanje

konfiguracije projekta je uporabljen razširljivi označevalni jezik (ang., Extensible Markup Language, XML). Za dodajanje ali spreminjanje sledilnikov ter sekvenc, je XML dokument potrebno ustrezno urediti. Primer 2.1 prikazuje enostavno konfiguracijo projekta z uporabo XML jezika. Za dodajanje sledilnika v projekt uporabimo element `tracker`, ki mu kot atribute podamo pot do izvršljive datoteke sledilnika `command`, unikatno identifikacijsko ime `id` ter ime sledilnika `title`. V projekt je potrebno vključiti še podatke o sekvencah, na katerih želimo sledilnik testirati. To storimo z elementom `include`, ki mu podamo pot do XML dokumenta, s podatki o sekvencah.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project xmlns="http://www.vicos.si/traxtor" id="">
  <tracker command="/home/piadph/asms/meanshift" id="meanshift" title="meanshift">
    <property name="deterministic" rel="extension">true</property>
  </tracker>
  <include src="/home/piadph/traxtor-results/vot2013.xml"/>
  <experiment id="optimization_1" type="optimization">
    <property name="runtime.values">false</property>
    <property name="gui.visible">true</property>
    <property name="generator.tracker">:meanshift</property>
    <property name="generator.sequences">vot2013:face;vot2013:iceskater</property>
  </experiment>
</project>
```

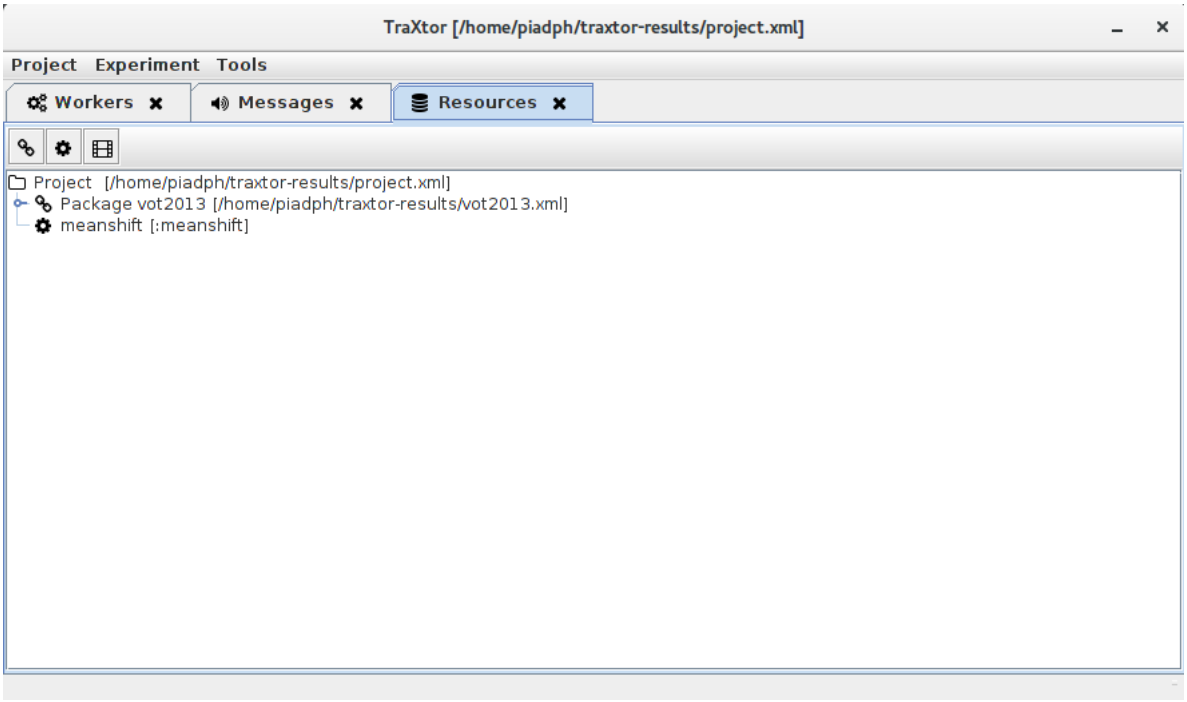
Primer 2.1: Enostavna konfiguracija projekta v obliki XML.

Primer 2.2 prikazuje XML strukturo dokumenta, kjer so opisane sekvence. V primeru sta prikazani le dve in za dodajanje novih, dokument le dopolnimo po izgledu v primeru. Za vsako sekvenco dodamo nov element `sequence`, kateremu kot atribute podamo ime sekvence `id` ter pot do direktorija `source path`, kjer se nahajajo slike.

```
<package xmlns="http://www.vicos.si/traxtor" id="vot2013">
<property name="author">Luka Cehovin</property>
<property name="title">VOT2013 sequence set</property>
<sequence id="bicycle">
    <source path="/home/piadph/vot2013/bicycle/" />
</sequence>
<sequence id="bolt">
    <source path="/home/piadph/vot2013/bolt/" />
</sequence>
</package>
```

Primer 2.2: XML dokument opisa sekvenc.

Če je projekt pravilno konfiguriran, se odpre okno, ki nam omogoča dodajanje in urejanje eksperimentov, njihovo izvedbo in shranjevanje rezultatov.



Slika 2.1: Primer TraXtor projekta.

Slika 2.1 prikazuje projekt Primer 2.1, kateri vsebuje le paket sekvenc in sledilnik meanshift. *Experiment* v orodni vrstici podpira dodajanje novega

eksperimenta, spreminjanje nastavitev že obstoječega ali njegovo odstranitev. Pri zagonu posameznega eksperimenta lahko v zavihku *Workers* spremljamo izvajanje in v zavihku *Messages* pregled sporočil ob kakršni koli napaki, ki se je zgodila med izvršbo sledilnika. Do obeh orodij lahko dostopamo preko elementa *Tools* v orodni vrstici.

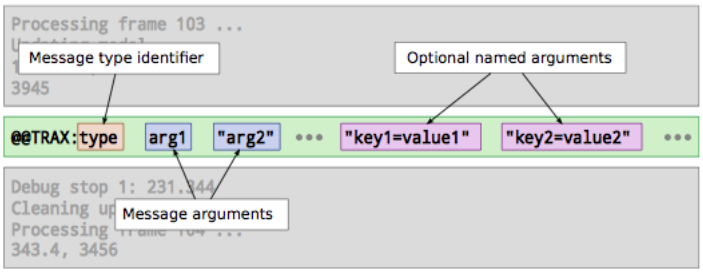
2.2 Protokol TraX

Protokol TraX [5, 45] je komunikacijski protokol, ki omogoča hitro in enostavno samodejno evaluacijo sledilnikov, ne glede na operacijski sistem ali uporabljen programski jezik. Z nekaj dodatnega dela ga je možno implementirati s katerim koli obstoječim sledilnim algoritmom. Omogoča komunikacijo med procesi sledilnika in procesi kontrolnega sistema, ki poteka preko standardnega komunikacijskega kanala z izmenjavo vrstičnih (ang., line-based) sporočil. Za lažje razumevanje so v nadaljevanju prevzeti osnovni izrazi tipičnega protokola, pri čemer je strežnik izbran sledilnik, odjemalec pa aplikacija TraXtor. Z razliko od tradicionalnih protokolov, kjer strežnik nenehno nudi podporo odjemalcem, v primeru protokola Trax, odjemalec zažene strežnik in ustvari se proces, preko katerega poteka izmenjava sporočil. V naslednjih podpoglavjih je podrobneje opisana oblika sporočila, stanja odjemalca in strežnika ter integracija protokola z obstoječim sledilnim algoritmom.

2.2.1 Oblika sporočila

Tako strežnik kot odjemalec uporabljata enako obliko sporočila, ki se začne in konča z novo vrstico. To pomeni, da je eno sporočilo ena vrstica, ločena od preteklega in prihodnjega toka podatkov. Slika 2.2 prikazuje strukturo tipičnega sporočila TraX, za komunikacijo med strežnikom in odjemalcem. S predpono "@@TRAX" ločimo med sporočili protokola in ostalimi sporočili, ki se lahko pojavijo na standardnem vhodu. Sledi ji vrsta sporočila *type* in argumenti ločeni s presledki. Argumente delimo na obvezne in neobvezne.

Tipu sporočila najprej sledijo obvezni argumenti, nato pa še argumenti, s katerimi lahko pošljemo dodatne podatke.



Slika 2.2: Ilustracija tipičnega sporočila TraX [45].

2.2.2 Vrste sporočil protokola

Sporočila, preko katerih odjemalec in strežnik komunicirata, morajo zadostiti, ne glede na preprostost protokola, točno določenim pravilom, saj nepravilna uporaba povzroči prekinitev povezave. V tem podpoglavju so opisani dovoljeni tipi sporočila in njegovi argumenti.

hello S tem sporočilom se strežnik predstavi odjemalcu. Dodatni argumenti niso potrebni, vendar lahko če želimo, pošljemo dodatne podatke. Ti argumenti so:

- trax.version (celo število) - trenutna verzija protokola.
- trax.name (niz) - ime sledilnika.
- trax.identifier (niz) - verzija trenutne implementacije sledilnika.
- trax.image (niz) - trenutno podprte oblike slik in
- trax.region (niz) - trenutno podprte oblike regije.

initialize Odjemalec s tem sporočilom pošlje zahtevo za inicilalizacijo sledilnika. Sporočilo mora vsebovati podatke o obliki slike in regije, ki morata biti podprti tudi s strani strežnika.

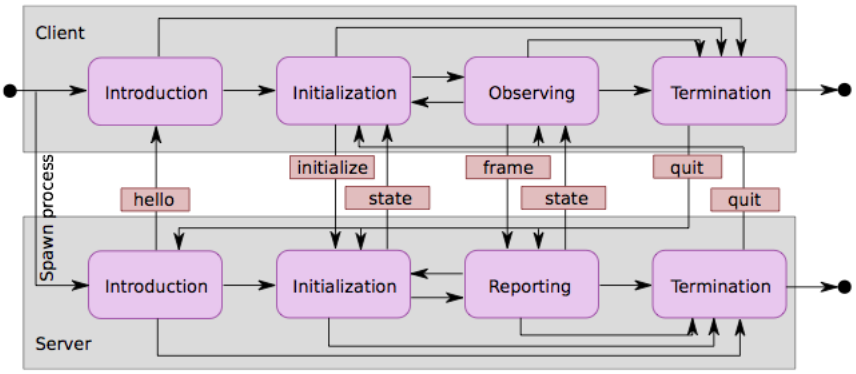
frame S tem sporočilom odjemalec pošlje podatke o novi sliki.

state Stežnik to sporočilo uporablja za odgovor odjemalcu s podatki o novo izračunani regiji.

quit Tako strežnik kot odjemalec lahko to sporočilo uporabita za prekinitev povezave.

2.2.3 Stanja odjemalca in strežnika

Potek komunikacije med strežnikom in odjemalcem lahko enostavno predstavimo s štirimi stanji. Slika 2.3 prikazuje prehode med stanji in sporočila, ki se pošiljajo med odjemalcem in strežnikom. V nadaljevanju so podrobneje opisana stanja in komunikacija, za prehod med njimi.



Slika 2.3: Grafični prikaz stanj strežnika in odjemalca ter potek pošiljanja sporočil med njima. [45]

Stanja odjemalca

1. **Predstavitev**: Pred zavzetjem tega stanja, odjemalec ustvari komunikacijski kanal, preko katerega bo potekala komunikacija s strežnikom. Premakne se v začetno stanje in čaka na odgovor strežnika. V primeru, da prejme sporočilo `hello` nadaljuje v naslednje stanje. V nasprotnem primeru zaključi povezavo.

2. **Inicializacija**: Pri inicializaciji odjemalec strežniku pošlje začetno sliko ter regijo za sledenje objekta, se postavi v stanje opazovanja in čaka na odgovor strežnika.

3. **Opazovanje**: Odjemalec čaka na sporočilo strežnika. V primeru odgovora `state`, nadaljuje s procesiranjem podatkov in se premakne v ustrezno stanje (inicializacija, opazovanje ali prekinitev).

4. **Prekinitev**: Prekinitev je možna s pošiljanjem zahteve `quit`.

Stanja strežnika

1. **Predstavitev**: Odjemalcu se predstavi s sporočilom `hello`.

2. **Inicializacija**: Strežnik čaka na sporočilo odjemalca (`initialize` ali `quit`). Ob zahtevi `initialize` se inicializira sledilnik in strežnik se premakne v stanje poročanja.

3. **Poročanje**: Strežnik čaka na nadaljnja sporočila odjemalca, katera so lahko `frame` (sledilnik pridobi naslednjo zaporedno sliko), `initialize` (sledilnik se znova inicializira) ali `quit` (strežnik se premakne v stanje prekinitve).

4. **Prekinitev**: Strežnik pošlje sporočilo `quit` ter konča z izvajanjem.

2.2.4 Oblike podatkov

Podatke o regiji in sliki si odjemalec in strežnik pošiljata preko sporočil. Trenutno je podprtih le nekaj oblik podatkov. Podprti obliki **regije** sta pravokotnik in poligon. Pravokotnik je določen s štirimi parametri - x in y koordinati zgornjega levega kota ter širino in višino pravokotnika. Poligon pa je definiran z najmanj šestimi točkami, predstavljenimi z x in y koordinatami. Podprti obliki **slik** sta JPEG in PNG.

2.2.5 Integracija protokola

Sledilni algoritem, ki ga želimo uporabiti, je potrebno predelati tako, da bo možna komunikacija z odjemalcem - aplikacijo TraXtor. Uporabljen je lahko poljuben programski jezik, le paziti je potrebno, da je uporaba protokola pravilna. V Algoritmu 1 je prikazan postopek za implementacijo strežnika z obstoječim sledilnim algoritmom. Implementacija protokola je možna tudi z odprtokodno knjižnico [5], ki trenutno podpira sledilnike implementirane v jezikih C++, C, Python in Matlab. Knjižnica je napisana v programskem jeziku C in v večini primerov omogoča lažjo in hitrejšo integracijo.

Na začetku vzpostavimo komunikacijski kanal, preko katerega si odjemalec TraXtor in stežnik (slednilnik) izmenjujeta sporočila. Dokler eden izmed njih ne zaključi povezave s sporočilom `quit`, bodisi zaradi uspešnega zaključka ali prekinitve med sledenjem, strežnik čaka na ukaze odjemalca. Ob sporočilu `initialize` iz sporočila preberemo podatke o regiji in trenutni sliki ter kreiramo model, katerega želimo slediti. Odjemalcu odgovorimo s sporočilom state, katero vsebuje podatke o regiji. Sporočilo `frame` pošlje zahtevo za izračun lokacije regije v sliki s poljubnim algoritmom. Strežnik po uspešnem izračunu odgovori odjemalcu z novo izračunano lokacijo. Po uspešnem ali neuspešnem sledenju se zapre komunikacijski kanal.

Algoritem 1 Primer integracije strežnika v obstoječi sledilnik

```
Vzpostavitev komunikacijskega kanala
Dokler obstaja povezava z odjemalcem:
    Čakaj na sporočilo odjemalca
    Sporočilo initialize:
        Inicializacija sledilnika s pridobljeno sliko in regijo
        Odgovor odjemalcu s sporočilom state in opisom regije
    Sporočilo frame:
        Izračun nove lokacije regije s poljubnim sledilnim algoritmom
        Odgovor s sporočilom state, ki vsebuje opis izračunane regije
    Sporočilo quit
        Prekini povezavo z odjemalcem
    Počisti vire sledilnika
Zapri komunikacijski kanal
```

2.3 Ocenjevanje sledilnika

Pri ocenjevanju sledilnikov, enotnega pristopa za določanje uspešnosti ni. Izmed metod so najbolj ustaljene mere preseka regije, napaka centralne točke, število odpovedi, dolžina sledenja do prve odpovedi sledilnika in ocena F [21]. Za natančnost sledilnika se velikokrat uporablja mera preseka regij, robustnost pa s številom neuspelih sledenj objekta. Odpoved sledilnika pomeni, da je bila meja natančnosti pri izračunu lokacije objekta v neki točki presežena. Z določanjem meje, v nadaljevanju τ, lahko vplivamo na potek sledenja. Večanje meje ponavadi vodi k večjemu številu odpovedi sledilnika, z manjšanjem pa velikokrat dobimo slabo natančnost. V raziskovalnih delih [9, 10] se običajno uporablja $\tau = 0.1$, predlagana pa je tudi precej visoka meja [19], kjer je $\tau = 0.5$. Če nas zanimajo le najbolj očitne odpovedi sledilnika, mejo nastavimo na najnižjo možno $\tau = 0$ [8]. Za realnejšo oceno moramo upoštevati tako mero robustnosti kot mero natančnosti ali pa se poslužiti hibridih metod. Dobra lastnost hibridnih metod je, da je primerjanje

med sledilniki lažje, saj imamo le eno število, kar pa velikokrat pomeni težjo interpretacijo rezultatov, saj nimamo vpogleda v potek sledenja. V nadaljevanju sta izpostavljeni meri preseka regije, ki določa natančnost sledilnika ter število neuspelih sledenj, ki nam pove kako robusten je sledilnik. Objekt, ki mu sledimo lahko generalno opišemo s stanjem Λ v sekvenci dolžine N

$$\Lambda = \{(A_t, \mathbf{x}_t)\}_t^N,$$

ki je določeno s centrom $x_t \in R^2$ regije A_t v času t [8].

2.3.1 Natančnost in robustnost

Izpostavljena metoda preseka regije velja za eno boljših [11] in je bila uporabljena tudi v sklopu naloge. Natančnost izračunane lokacije objekta je izmerjena preko enačbe (2.1) in prestavlja presek regije v trenutni sliki i_t. Za vsako sliko v sekvenci dolžine N, sledilnik poda izračunano lokacijo in velikost detektirane regije A_t^T.

$$\Phi(\Lambda^G, \Lambda^T) = \{\phi_t\}_{t=1}^N, \phi_t = \frac{A_t^G \cap A_t^T}{A_t^G \cup A_t^T}, \tag{2.1}$$

kjer je A_t^G točna lokacija objekta. Slika 2.4 prikazuje primer sledenja objekta na anotirani javno dostopni sekvenci [36]. Na sliki je viden zelen pravokotnik, ki predstavlja točno lokacijo objekta (ang., ground-truth) in rdeč pravokotnik, katerega lokacijo je napovedal sledilnik. Z modrim pravokotnikom je označen presek obeh regij. V primerjavi s priljubljeno mero računanja napake centralne točke regije, se pri računanju preseka upošteva tudi velikost okna [8]. Napačna detekcija v tem primeru ne vrne prevelike napake, tudi če okno zdrsne daleč v ozadje. Rezultat je v tem primeru enak nič. S tem, ko odpoved sledilnika ne povzroči velike napake, iz rezultata težko določimo ali je bilo sledenje zares uspešno. Pri računanju končnega rezultata se skupni seštevek skozi celotno sekvenco povpreči s številom slik v sekvenci.

$$\Phi = \frac{1}{N} \sum_{t=1}^{N} \phi_t \tag{2.2}$$

Slika 2.4: Primer sledenja objekta. Prikazana je točna regija (zelena), napovedana regija (rdeča) in prekrivanje regije (modra).

Za zagotavljanje kar se da realnih rezultatov moramo upoštevati tudi robustnost sledilnika, saj iz povprečne prekritosti regije težko vidimo kakšen je bil potek sledenja.

2.4 Algoritem mean shift

Postopek srednjega premika (ang., Mean Shift, MS) je neparametričen algoritem, ki z vzponom po gradientu iterativno išče maksimum funkcije gostote porazdelitvene verjetnosti (ang., probability density function, pdf). Predstavila sta ga Fukunaga in Hostetler, leta 1975 [20]. Kasneje, leta 1999, ga je Comaniciu vpeljal v področje računalniškega vida [15,17] in predlagal njegovo aplikacijo za sledenje objektom. Sledenje se izvaja z minimizacijo razdalje oziroma razlike med dvema funkcijama gostote porazdelitvene verjetnosti. Funkciji nad katerima se izvaja primerjava sta predstavljeni s histogramom modela objekta, ki ga sledimo ter histogramom kandidata. Osnovni algoritem ne upošteva orientacije in spremembe sledenega objekta, na kar je opozoril že Comaniciu [17] in so bile kasneje predlagane metode, ki so spremembe upoštevale [16, 34, 44].

2.4.1 Sledilnik Mean Shift

Osnovna ideja sledenja objektu z uporabo algoritma mean shift je preprosta. Inicializiramo model, katerega iščemo v vseh naslednjih zaporednih slikah v

sekvenci. Največkrat je model predstavljen kot barvni histogram, možno pa je uporabiti tudi histograme, ki opisujejo intenziteto ali robove [18]. V naslednji sliki nato iščemo kandidata, kjer je podobnost med funkcijama največja oziroma je razdalja med njima najmanjša. Ker z uporabo histograma izgubimo informacijo o okolici objekta, se za boljše ujemanje med kandidati in modelom vpelje masko - simetrično jedro ali kombinacijo simetričnih jeder, ki omogočajo upoštevanje različnih lastnosti okolice in modela.

Na začetku določimo barvni prostor in m število skupin (ang., bin) barvnega histograma. V začetni sliki izračunamo histogram referenčnega modela,

$$\hat{\mathbf{q}} = \{\hat{\mathbf{q}}_u\}_{u=1\ldots m} \sum_{u=1}^{m} \hat{\mathbf{q}}_u = 1 \tag{2.3}$$

in na vsaki naslednji sliki poiščemo kandidata, opisanega s histogramom

$$\hat{\mathbf{p}}(\mathbf{y}) = \hat{\mathbf{p}}_u(\mathbf{y})_{u=1..m} \sum \hat{\mathbf{p}}_u = 1. \tag{2.4}$$

Verjetnost pojavitve značilke $u \in \{1...m\}$ je definirana s funkcijo

$$\hat{\mathbf{q}}_u = C \sum_{i=1}^{N} k(\| \mathbf{x}_i^* \|^2) \delta_u(b(\mathbf{x}_i^*)), \tag{2.5}$$

kjer je $\mathbf{x}_i$ vrednost slikovnega elementa v sliki in $b(\mathbf{x}_i)$ skupina v histogramu, kateri pripada vredost $\mathbf{x}_i$ ter je δ Kroneckerjeva delta funkcija, določena z enačbo (2.7). $k(x)$ predstavlja izbrano jedro in C normalizacijsko konstanto. Kandidat na lokaciji $\mathbf{y}$ je opisan s funkcijo gostote porazdelitvene verjetnosti $p(\mathbf{y})$, kjer je h skalirni parameter (tipično $h = 0.3$ [40]).

$$\hat{\mathbf{p}}_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k(\| \frac{\mathbf{y} - \mathbf{x}_i}{h} \|^2) \gamma[b(\mathbf{x}_i) - u], \tag{2.6}$$

$$\delta_u = \begin{cases} 0, & u \neq 0 \\ 1, & u = 0 \end{cases}, \tag{2.7}$$

$$C_h = \frac{1}{\sum_{i=1}^{n_h} k(\| \frac{\mathbf{y}-\mathbf{x}_i}{h} \|^2)}. \tag{2.8}$$

Podobnost med referenčnim modelom $\hat{q}$ in kandidatom $\hat{p}(y)$ določimo z minimizacijo Hellingerjeve razdalje $H(\cdot,\cdot)$

$$H(\hat{p}(y),\hat{q}) = \sqrt{1-\rho[\hat{p}(y),\hat{q}]}, \tag{2.9}$$

kjer je ρ Bhattacharyya koeficient

$$\rho[\hat{p}(y),\hat{q}] = \sum_{u=1}^{m}\sqrt{\hat{p}_u(y)\hat{q}_u}. \tag{2.10}$$

V vsaki zaporedni sliki se sledenje začne v točki $\hat{y}_0$ iz prejšnjega okna in se s pomočjo vzpona po gradientu premikamo na novo lokacijo $\hat{y}_1$.

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h}\mathbf{x}_i w_i g(\| \frac{\hat{\mathbf{y}}_0-\mathbf{x}_i}{h} \|^2)}{\sum_{i=1}^{n_h} w_i g(\| \frac{\hat{\mathbf{y}}_0-\mathbf{x}_i}{h} \|^2)}. \tag{2.11}$$

Uteži w_i so izračunane po enačbi

$$w_i = \sum_{u=1}^{m}\sqrt{\frac{\hat{\mathbf{q}}_u}{\hat{\mathbf{p}}_u(\hat{\mathbf{y}}_0)}}\delta[b(\mathbf{x}_i)-u]. \tag{2.12}$$

in je $g(x) = -k'(x)$ odvod funkcije $k(x)$, za katero velja, da obstaja za vse $x \geq 0$, razen za končno množico točk.

2.4.2 Adaptacija velikosti

Za adaptacijo velikosti objekta skozi sekvenco, sleden objekt opišemo z eliptično regijo $\frac{(x_i^1)^2}{a^2}+\frac{(x_i^2)^2}{b^2}<1$, kjer sta a in b polosi elipse. Lokacijo slikovnega elementa v sliki z N številom elemetov opišemo z $y=(y^1,y^2)^T$, $x_i=(x_i^1,x_i^2)^T$. Izračunamo verjetnost pojavitve značilke $u \in \{1...m\}$ referenčnega modela

$$\hat{q}_u = C\sum_{i=1}^{N} k(\frac{(x_i^1)^2}{a^2}+\frac{(x_i^2)^2}{b^2})\delta[b(x_i^*)-u], \tag{2.13}$$

kjer je C normalizacijska konstanta. Izračun verjetnosti pojavitve značilke $u \in \{1...m\}$ kandidata pa izračunamo po enačbi

$$\hat{p}_u(y,h) = C_h\sum_{i=1}^{N} k(\frac{(y^1-x_i^1)^2}{a^2h^2}+\frac{(y^2-x_i^2)^2}{b^2h^2})\delta[b(x_i)-u], \tag{2.14}$$

kjer je $k(x)$ poljubno simetrično jedro in je δ, Kroneckerjeva delta funkcija (2.7). Z h označimo vrednost skalirnega faktorja. Izračun aproksimalne vrednosti C_h lahko določimo kot $C_h \approx C\frac{1}{h^2}$ in je za katerikoli dve vrednosti h_0, h_1 $C_{h_1} \approx C_{h_0}\frac{h_0^2}{h_1^2}$ [40]. Z aproksimacijo C_h v okolici h_0 s pomočjo Hellingerjeve razdalje (2.9) izračunamo podobnost med referenčnim modelom in kandidatom

$$\begin{aligned}\rho[\hat{p}(y,h),\hat{q}] &\approx \hat{\rho}(y,h)\\ &= \sum_{u=1}^{m}\sqrt{C_{h_0}\frac{h_0^2}{h^2}\sum_{i=1}^{N}k(\frac{(y^1-x_i^1)^2}{a^2h^2}+\frac{(y^2-x_i^2)^2}{b^2h^2})\delta[b(x_i)-u]\hat{q}_u}.\end{aligned} \tag{2.15}$$

Določimo

$$w_i = \sum_{u=1}^{m}\sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{y}_0,h_0}})\delta[b(x_i)-u] \tag{2.16}$$

$$G = \sum_{i=1}^{N} w_i g(\frac{(\hat{y}_0^1-x_i^1)^2}{a^2h_0^2}+\frac{(\hat{y}_0^2-x_i^2)^2}{b^2h_0^2}) \tag{2.17}$$

$$m_k(\hat{y}_0,h_0) = \frac{\sum_{i=1}^{N} x_i w_i g(\frac{(\hat{y}_0^1-x_i^1)^2}{a^2h_0^2}+\frac{(\hat{y}_0^2-x_i^2)^2}{b^2h_0^2})}{G} - \hat{y}_0 \tag{2.18}$$

kjer je $m_k(\hat{y}_0,h_0) = (m_k^1(\hat{y}_0,h_0), m_k^2(\hat{y}_0,h_0))^T$.
Mean shift premik na lokacijo y in skalirni faktor h sta tako določena z

$$\hat{y}_1^1 = \frac{1}{a^2}m_k^1(\hat{y}_0,h_0) + \hat{y}_0^1 \tag{2.19}$$

$$\hat{y}_1^2 = \frac{1}{b^2}m_k^2(\hat{y}_0,h_0) + \hat{y}_0^2 \tag{2.20}$$

$$\begin{aligned}h_1 = [1 &- \frac{\sum_{i=1}^{N} w_i k(\frac{(\hat{y}_0^1-x_i^1)^2}{a^2h_0^2}+\frac{(\hat{y}_0^2-x_i^2)^2}{b^2h_0^2})}{G}]h_0\\ &+ \frac{1}{h_0}\frac{\sum_{i=1}^{N} w_i(\frac{(\hat{y}_0^1-x_i^1)^2}{a^2}+\frac{(\hat{y}_0^2-x_i^2)^2}{b^2})g(\frac{(\hat{y}_0^1-x_i^1)^2}{a^2h_0^2}+\frac{(\hat{y}_0^2-x_i^2)^2}{b^2h_0^2})}{G}.\end{aligned} \tag{2.21}$$

2.5 Optimizacijske metode

Za določitev optimalnega nabora vrednosti parametrov, kjer je ocena sledenja najboljša, moramo sledilnik testirati na vseh možnih kombinacijah vrednosti. Z majhnim številom parametrov in vrednosti, ki jih lahko zavzamejo, to ne predstavlja težave, saj kombinacij ni veliko. Z večanjem problema se zelo hitro poveča tudi število kombinacij, kar privede do prevelike računske in časovne zahtevnosti reševanja. Zaradi tega vpeljemo optimizacijske metode, ki lahko privedejo do enako dobrih rezultatov ali pa se jim zelo približajo. Glede na vrsto problema moramo izbrati primerno metodo, ki je odvisna od kriterijske funkcije, vrsto optimizacije in rešljivosti problema.

2.5.1 Določitev problema

Imamo množico Ψ dopustnih rešitev in definirano funkcijo P nad množico [1]. Zvezo med njima opišemo z

$$P : \Psi \to \bar{\mathbb{R}}, \tag{2.22}$$

kjer je $\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. Naloga optimizacije je minimizirati ali maksimizirati vrednost kriterijske funkcije P, iščemo $Min(\Psi, P)$ oziroma $Max(\Psi, P)$. Določimo tudi množico Σ, s katero opišemo kdaj je naloga rešena. To je lahko množica več rešitev, ene rešitve ali le njene vrednosti.

$$Min(\Psi, P) = \{x \in \Psi : \forall y \in \Psi \in \Psi : P(y) \geq P(x)\} \tag{2.23}$$

in je

$$Max(\Psi, P) = Min(\Psi, -P). \tag{2.24}$$

Za vsak par $x, y \in Min(\Psi, P)$ velja $P(x) \geq P(y)$ in $P(y) \geq P(x)$, iz česar sledi da je $P(x) = P(y)$. Enakost pomeni, da imajo vsi elementi množice rešitev $\Sigma = Min(\Psi, P)$ enako vrednost kriterijske funkcije.
V primeru optimizacije parametrov iščemo kombinacijo vrednosti, ki bo ustrezala optimizacijski nalogi iskanja maksimuma ali minimuma kriterijske funkcije, iščemo (Ψ, P, Max) ali (Ψ, P, Min). Problem spada v naloge diskre-

tne optimizacije, kjer je množica Ψ podmnožica ene izmed množic kombinatoričnih konfiguracij (permutacije, kombinacije, razbitja,...) in je moč množice Ψ končna ali števno neskončna. Ko imamo problem optimizacije definiran, se lahko lotimo iskanja metode reševanja problema. Metoda izčrpnega preiskovanja, kjer preverimo vse možne kombinacije, sicer vedno najde globalni optimum, vendar je ta metoda največkrat računsko in časovno prezahtevna. S tem razlogom vpeljemo metodo lokalne optimizacije, ki bo efektivno preiskala izbrane kombinacije in se čim bolje približala optimalni rešitvi.

Postopek za lokalno optimizacijo lahko razdelamo na:

1. **Določitev sosednosti.** Sosednost predstavlja število točk oziroma vzorcev v danem intervalu. Gostejša kot je porazdelitev točk v intervalu, večja je verjetnost, da ne bomo obtičali v lokalnem optimumu. Z večanjem števila točk, pa se poveča tudi število kombinacij. Poiskati je potrebno ravnovesje med velikostjo sosednosti in računsko zahtevnostjo evaluacije.

2. **Izbira začetnih vrednosti parametrov.** Z dobro izbiro začetnih vrednosti se lahko izognemo lokalnemu optimumu. Vrednosti parametrov lahko izberemo sami ali uporabimo metodo, ki to naredi namesto nas. Metoda za iskanje najboljšega nabora vrednosti začetnih vrednosti, je lahko velikokrat sama po sebi zelo zahtevna naloga.

3. **Ustavitev.** Ustavitev iskanja optimuma funkcije lahko definiramo kot število dovoljenih iteracij ali pa uvedemo kriterij ustavitve. V tem primeru se postopek iskanja izvaja dokler prihaja do izboljšav.

4. **Izberemo naslednjo rešitev.** Naredimo premik v smer rešitve, ki jo predlaga izbrana preiskovalna metoda in postopek ponovimo.

2.5.2 Preiskovalne metode

Ko imamo optimizacijski problem določen, izberemo metodo preiskovanja prostora. Za probleme kombinatorične optimizacije se velikokrat uporabijo hevristične metode, ki sicer ne zagotavljajo optimalnosti, vendar se zelo dobro obnesejo v praksi. Izmed enostavnejših metod je metoda vzpona po hribu (ang., Hill-Climbing, HC), katero lahko klasificiramo v štiri glavne različice [38].

Vzpon v katerokoli smeri (ang., Any-Ascent Hill-Climbing, AHC) se premakne v smer, kjer je rezultat kriterijske funkcije večji ali enak trenutnemu.

Stohastični vzpon (ang., Stochastic Hill-Climbing) je različica AHC, kjer se smer vzpona določi naključno.

Prvi vzpon (ang., First-Ascent Hill-Climbing, FAHC) se vzpne v smer prve izboljšave, ki jo najde.

Najstrmejši vzpon (ang., Steepest-Ascent Hill-Climbing, SAHC) sistematično preveri okolico trenutno najboljše rešitve in se premakne v smer, kjer je bila izboljšava največja.

Pri vseh naštetih različicah je možnost, da se metoda ustavi v lokalnem optimumu in zgreši globalnega. Obstaja kar nekaj metod, ki se temu lahko uspešno izognejo. Nekaj izmed teh je metoda ohlajanja (ang., Simulated Annealing, SA), evolucijski algoritmi (ang., Evolutionary Algorithms, EAs) in metoda prepovedanih smeri (ang., Tabu Search, TS).

Poglavje 3

Sistem za optimizacijo

V tem poglavju je predlagan in opisan pristop za iskanje najboljšega nabora vrednosti parametrov ter je predstavljena nova metoda za ocenjevanje sledilnikov. V Podpoglavju 3.1 je opisan dodan eksperiment in njegova uporaba. Postopek dodajanja novega sledilnika v sistem, je opisan v Podpoglavju 3.2. Nova hibridna metoda, ki je bila dodana v sistem je podrobneje opisana v Podpoglavju 3.4. Na koncu, v Podpoglavju 3.5, je predlagan grafični prikaz rezultatov in njegova interpretacija.

3.1 Razširitev sistema TraXtor

Obstoječi sistem za ocenjevanje sledilnikov, podrobeje opisan v Poglavju 2, Podpoglavje 2.1, je aplikacija, implementirana v programskem jeziku Java. Omogoča testiranje, ocenjevanje in vizualno evaluacijo sledilnikov. Ker podpira le evaluacijo sledilnika brez nadaljnje obdelave ali analize rezultatov, je bilo potrebno sistem razširiti tako, da bo podpiral samodejno določanje optimalnega nabora vrednosti parametrov glede na rezultat uspešnosti sledenja. V tem delu bo podrobneje opisan le razširjeni del sistema, njegova uporaba ter možnost dodajanja poljubnega sledilnega algoritma.

3.1.1 Optimizacijski eksperiment

Javanski projekt TraXtor je logično razdelan v skupine po njihovi namembnosti. Paketi, ki so bili za razširitev pomembni in kjer se dodani razredi nahajajo so *si.vicos.tracking.desktop.panels* in *si.vicos.tracking.experiment*. Ostali paketi in razredi ostanejo enaki in so v dodanih razredih le uporabljeni.

V sistem so bili dodani štirje novi razredi in sicer:

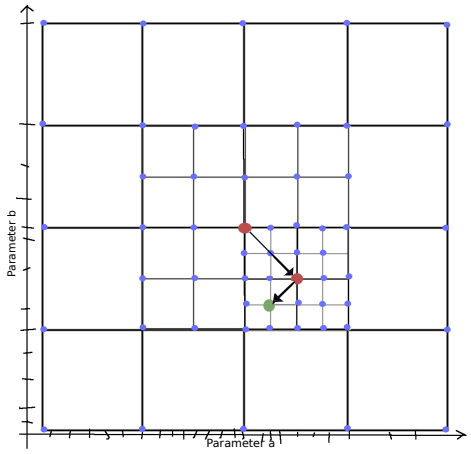
- `OptimizationExperiment` (*si.vicos.tracking.experiment*),
- `OptimizationExperimentPanel` (*si.vicos.tracking.desktop.panels*),
- `OptimizationParameters` (*si.vicos.tracking.experiment*) in
- `OptimizationTask` (*si.vicos.tracking.experiment*).

Razred `OptimizationExperimet` določa izgled eksperimenta in uporabniku omogoča spreminjanje začetnih vrednosti sledilnika ter okoljskih spremenljivk. Metoda preiskovanja prostora je implementirana v razredu `OptimizationExperimentPanel`. Z razlogom, da lahko eksperiment poženemo brez ročnega nastavljanja, je bil kreiran razred `OptimizationParameters`, kateri vsebuje privzete vrednosti parametrov sledilnika in okoljskih spremenljivk eksperimenta. Privzete vrednosti se tako upoštevajo le tam, kjer uporabnik ni navedel vrednosti parametrov. `OptimizationTask` je razred, ki določa potek sledenja na posamezni sekvenci.

3.1.2 Metoda iskanja parametrov

Iskanje optimalnega nabora vrednosti parametrov je problem kombinatorične optimizacije, podrobneje opisane v Poglavju 2, Podpoglavje 2.5. Uporabljena je bila metoda najstrmejšega vzpona po hribu (ang., Steepest-Ascent Hill-Climbing, SAHC), ki je različica vzpona po hribu (ang., Hill-Climbing, HC). Različica SAHC se od HC razlikuje v tem, da pregleda okolico začetnih vrednosti in izbere najboljšo. Osnovna verzija HC izbere prvo rešitev, ki je boljša

od trenutne. S tem, ko metoda vedno izbere najboljšo rešitev se zmanjša verjetnost, da obtičimo v lokalnemu maksimumu. Metoda deluje tako, da izberemo začetni nabor vrednosti in glede na sosednost določimo število vzorcev, ki bodo evaluirani. Po izračunu se premaknemo v smer, kjer je bila rešitev boljša od trenutne in ponovimo postopek dokler ne presežemo števila iteracij. Za doseg optimalne rešitve je število iteracij odvisno od začetne sosednosti in širine intervala vrednosti, ki jih parameteri lahko zavzamejo.

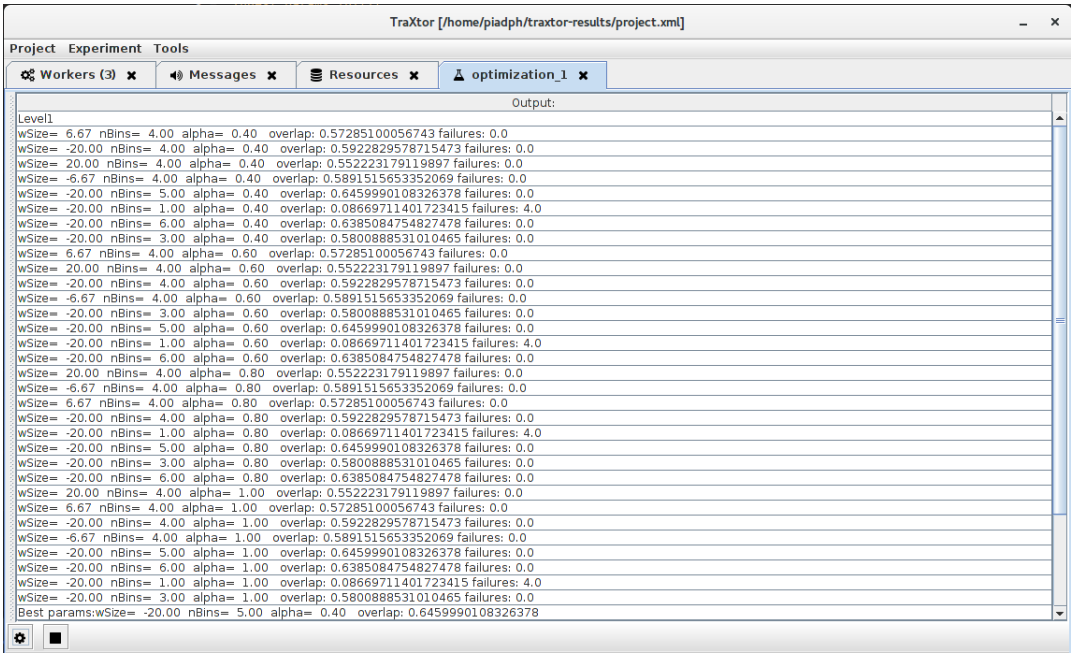


Slika 3.1: Skica preiskovanja prostora.

Slika 3.1 prikazuje iskanje optimalnega nabora vrednosti pri dveh parametrih. Na celotnem intervalu določimo sosednost, n in interval možnih vrednosti parametra razdelimo na n vzorcev. Modre točke predstavljajo kandidate, rdeče lokalne maksimume v vsaki iteraciji in zelena končni maksimum. V vsaki iteraciji interval preiskovanja zmanjšamo in pregledamo okolico.

3.1.3 Prikaz rezultatov

V trenutni verziji dodanega eksperimenta je prikaz rezultatov zelo enostaven. Za vsak nabor, se na zaslon v novo vrstico, izpišejo vrednosti parametrov in izračunana povprečna prekritost regije ter število odpovedi sledilnika. Na koncu se izpišejo še vrednosti, kjer je bila dosežena najboljša prekritost regije in vrednosti prametrov.



Slika 3.2: Trenutni prikaz rezultatov.

3.1.4 Uporaba eksperimenta

Optimizacijski eksperiment omogoča iskanje optimalnega nabora vrednosti parametrov, pri katerem je sledenje najbolj uspešno. Pri kreiranju novega eksperimenta se odpre konfiguracijsko okno, kjer izberemo sledilni algoritem, ki smo ga pravilno vključili v projekt (Poglavje 2, Podpoglavje 2.1.1). Izberemo sekvence, na katerih želimo evaluirati sledilni algoritem ter določimo mejo za katero privzamemo, da je sledilnik odpovedal. Z delta kriterijem določimo kako občutljiva je metoda preiskovanja prostora na spremembe vrednosti kriterijske funkcije. Brez ročnega nastavljanja vrednosti spremenljivk se program zažene s privzetimi nastavitvami za izbran sledilnik. Okoljske spremenljivke opisane v nadaljevanju, omogočajo spreminanje poteka delovanja preiskovalne metode.

Okoljske spremenljivke:

- **iter**: Število iteracij preiskovalne metode. Prva iteracija se izvede z

začetnima pogojema *rangeWidth* in *sampleCount*. V vsaki naslednji iteraciji se interval *rangeWidth* zmanjša za *rangeMult* in število točk v intervalu poveča za *sampleCount*. Privzeta vrednost *iter* je 2.

- **sampleCount**: Določitev začetne sosednosti. Intervali parametrov se razdelijo na *sampleCount* točk. Privzeta vrednost je 3.
- **sampleMult**: Koeficient s katerim povemo, za koliko se bo povečalo število točk v vsaki naslednji iteraciji. Vrednost 1 pomeni, da bo število točk *sampleCount* ostalo enako, koeficient 2 pa bo število točk podvojil. Privzeta vrednost je 2.
- **rangeMult**: Faktor za katerega se zmanjša interval vrednosti parametrov v vsaki naslednji iteraciji. Vrednost 1 bo ohranila velikost intervala *rangeWidth*, vrednost 0.5 ga bo zmanjšala za polovico. Privzeta vrednost je 0.5.
- **rangeWidth**: Začetna vrednost širine intervala parametrov sledilnika. Pri vrednosti 1 se bo celoten interval parametra razdelil na *sampleCount* točk. Predstavlja odstotek celotnega intervala in se upoštevajo le vrednosti okoli začetnih vrednosti sledilnika.

Spremenljivke sledilnika:
Spremenljivke sledilnika so določene za vsak sledilni algoritem posebej. Za primer sledilnika mean shift [17], so parametri `window`, `dim` in `alpha`. Parametri in primer uporabe sledilnika v eksperimentu so opisani v Podpoglavju 3.3.

3.2 Dodajanje novega sledilnika

Za uporabo sledilnika v optimizacijskem eksperimentu, je potrebno v sam javanski program dopisati nekaj vrstic kode. Trenutno, dodajanje sledilnika ni preveč uporabniku prijazna, vendar ni zahtevna. Za dodajanje sledilnika je potrebno dopolniti nekaj funkcij v razredu `OptimizationParameters`,

ki se nahaja v paketu *si.vicos.tracking.experiment*, ter dopolniti funkcijo `InitializeVariables` v razredu `OptimizationExperimentPanel`, ki se nahaja v paketu *si.vicos.trscking.desktop.panels*. Razred `OptimizationParameters` je bil ustvarjen predvsem zato, da je nadaljnje testiranje sledilnika enostavno in hitro. Nastavljanje privzetih vrednosti parametrov v samem programu omogoča, da lahko vrednosti spreminjamo brez potrebe po ponovnem prevajanju sledilnega algoritma.

V razred `OptimizationParameters` je potrebno dodati dva slovarja:

- `default_tracker_intervals` - kjer je ključ ime parametra in vrednost seznam zgornje in spodnje meje možnih vrednosti parametra. Sledi tip števila (double ali int), ki določa način vzorčenja vrednosti v intervalu.
- `default_tracker_parameters` - kjer je ključ ime parametra in vrednost, njegova vrednost. Uporabi se le v primeru, da uporabnik ne poda vrednosti ob zagonu eksperimenta.
- V ostale funkcije je potrebno dodati le še opcijo za dodani sledilnik.

V razredu `OptimizationExperimentPanel`:

- Funkcijo `InitializeVariables` dopolni z imenom ID svojega sledilnika in uporabi ustvarjene slovarje iz razreda `OptimizationExperiment`.

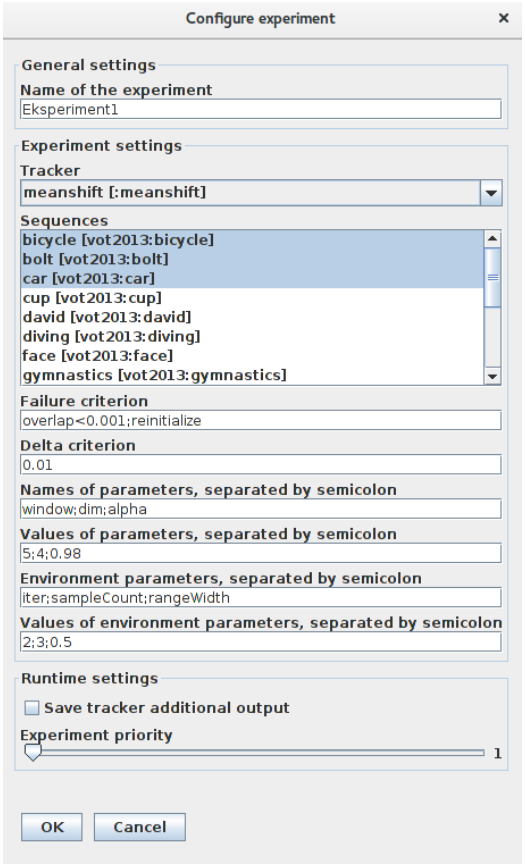
3.3 Primer uporabe

V tem delu je opisan način testiranja sledilnika v optimizacijskem eksperimentu na primeru sledilnika Mean Shift [17]. Glede na to, da je MS neparametričen algoritem, lahko na izboljšanje sledenja vplivamo le z nekaterimi zunanjimi parametri, kot so velikost sledilnega okna, število košev histograma in parametrom alpha [17]. Omenjeni parametri so bili uporabljeni v sklopu diplomske naloge in so podrobneje opisani v nadaljevanju.

- **alpha**: Objekt, ki ga sledimo, na slikah pogosto spreminja velikost, obliko ali barvo, kar lahko oteži sledenje. Spreminjanje objekta skozi sekvenco lahko obravnavamo tako, da histogram modela vsakič posodobimo, z na novo izračunanim histogramom objekta. Koeficient alpha določa, za koliko se bo histogram modela v vsaki zaporedni sliki spremenil glede na trenutni objekt. Vrednost 1 pomeni, da se upošteva le model izračunan ob inicializaciji, in se med sledenjem ne speminja. Z manjšanjem vrednosti se vse bolj upošteva trenutni histogram, kar lahko pripelje do napačne detekcije sledilnika, saj je model manj občutljiv na spremembe. Zgodi se lahko, da sledilnik detektira napačen objekt, kar vodi v odpoved sledilnika. Z večjimi vrednostmi koeficienta pa je objekt bolj občutljiv na spremembe objekta in v primeru, da objekt zelo spreminja izgled, se lahko hitro zgodi, da sledilnik odpove. V nalogi je vrednost *alpha* omejena na iterval $\{alpha \in \mathbb{R} : 0.85 \leq alpha \leq 1\}$, saj se je izkazalo, da pri primeru sledilnika MS, premajhne vrednosti ne pripeljejo k izboljšanju rezultata.

- **window**: Glede na objekt in ozadje v katerem se nahaja, lahko na uspešnost sledenja vplivamo z večanjem ali manjšanjem sledilnega okna. Če je okno preveliko, sicer najverjetneje ne bo prišlo do odpovedi sledilnika, zato pa bo natančnost lokacije objekta manjša. V osnovi MS algoritem dela bolje z nekoliko večjim sledilnim oknom. Z vrednostjo parametra window povemo, za koliko želimo zmanjšati ali povečati sledilno okno. Vrednosti so omejene na $\{window \in \mathbb{R} : -20 \leq window \leq 20\}$, ki predstavljajo odstotek originalnega sledilnega okna.

- **dim**: Model predstavljen kot barvni histogram lahko kvantiziramo z *dim* številom košev in s tem pridobimo različne informacije o regiji. Glede na število barv in karakteristik objekta, lahko model zelo dobro opišemo z dobro izbrano vrednostjo parametra. Parameter je omejen na vrednosti $\{dim \in \mathbb{N} : 2 \leq dim \leq 6\}$ in je število košev enako 2^{dim}.



Slika 3.3: Konfiguracija eksperimenta.

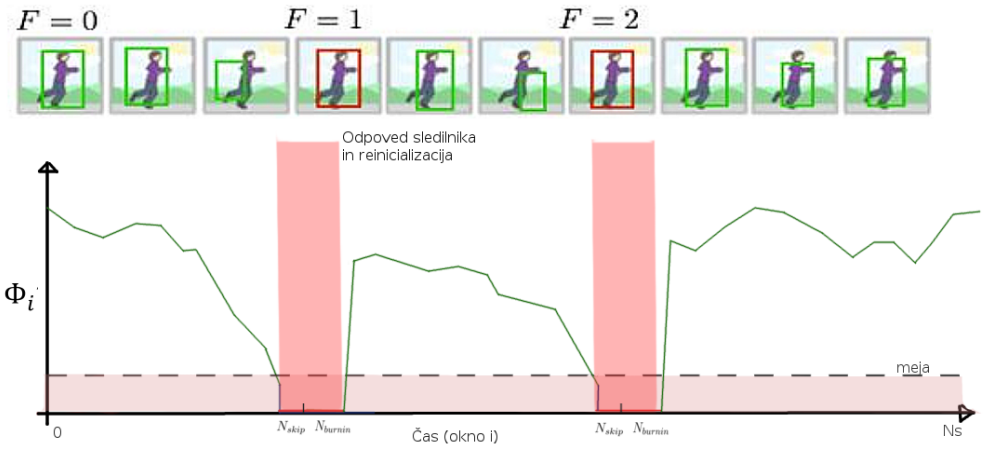
Slika 3.3 prikazuje primer konfiguracije eksperimenta. V *Experiment settings* izberemo sledilnik in sekvence, ki smo jih uspešno dodali v projekt (Poglavje 2, Podpoglavje 2.1.1). Določimo mejo odpovedi sledilnika (*overlap<0.001*) in povemo ali želimo reinicializacijo sledilnika (*reinitialize*). Izberemo delta kriterij (0.01), s katerim povemo, kdaj je vrednost zares boljša od trenutne. V primeru, da bi bila trenutna vrednost kriterijske funkcije $f(x_t) = 0.53$ in je kriterijska funkcija kandidata $f(x_{t+1}) = 0.54$, se rešitev ni izboljšala, saj je $|f(x_t) - f(x_{t+1})| \geq 0.01$.

Sledi mu izbor parametrov, ki jim želimo nastaviti začetne vrednosti. Imena parametrov morajo biti enaka imenom spremenljivk, ki smo jih uporabili v sledilniku in aplikaciji TraXtor. V primeru na sliki smo tako nastavili parametre *window=5, dim=4, alpha=0.98*. Sledilnik se bo izvedel s 5% večjim

sledilnim oknom, histogramom z 2^4 številom košev za vsak barvni kanal, ter alpha koeficientom, ki bo v vsaki zaporedni sliki, histogram modela prilagodil za 2%.
Z nastavitvijo okoljskih spremenljivk lahko spremenimo območje iskanja parametrov. V primeru konfiguracije na Sliki 3.3 smo nastavili število iteracij $iter = 2$, število začetnih točk $sampleCount = 3$ in širino intervala $rangeWidth = 1$. Ostale okoljske spremenljivke niso bile eksplicitno nastavljene, zato se bodo uporabile privzete vrednosti ($countMult$=2, $rangeMult$=0.5). V prvi iteraciji bo širina intervala polovica celotnega, ki se razdeli na $sampleCount$ točk. Po zagonu sledilnika na vseh kombinacijah dobimo optimalni nabor vrednosti parametrov prve iteracije. V drugi iteraciji se celotni interval okoli parametrov prve iteracije zmanjša za $rangeMult$ in število točk v zmanjšanem intervalu se poveča za $sampleMult$.

3.4 Ocenjevanje sledilnika

Za poenotenje ocenjevanja uspešnosti sledenja je predstavljena nova hibridna metoda, ki združuje mero robustnosti in mero natančnosti sledenja [26]. Mera vrne eno samo oceno, ki predstavlja okvirno pričakovano povprečno prekrivanje regije (ang., expected average overlap). Združevanje mere robustnosti in točnosti nam omogoča lažjo primerjavo uspešnosti sledilnikov in lažjo interpretacijo odvisnosti vpliva ene mere na drugo. Uporabljena mera pričakovanega povprečnega prekrivanja regij je prva metoda, ki upošteva reinicializaijo sledilnika med sledenjem. Reinicializacija sledilnika nam poda realnejšo oceno in omogoča podrobnejšo analizo poteka sledenja. Pri odpovedi sledilnika je velika verjetnost, da bo spet odpovedal takoj po reinicializaciji, zato preskočimo nekaj N_{skip} slik in ponovno inicializiramo model ter nadaljujemo s sledenjem. Obširnejše testiranje [26] je pokazalo, da kratkotrajne spremembe v sekvenci niso daljše od petih zaporednih slik, zato je $N_{skip} = 5$.



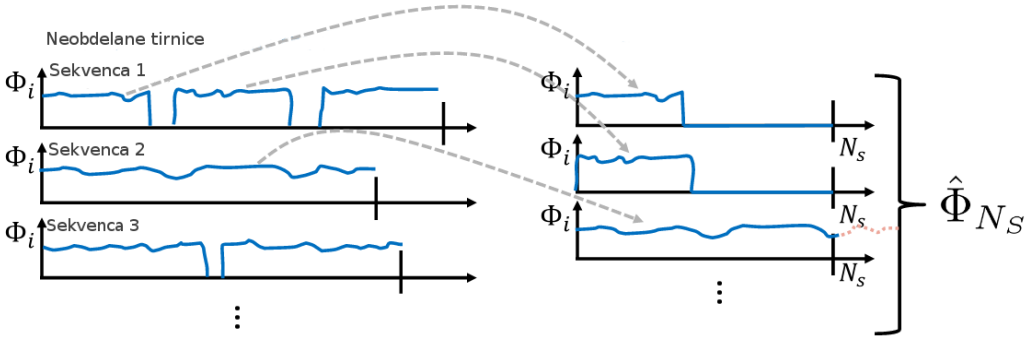
Slika 3.4: Potek sledenja [25].

Po ponovni inicializaciji modela je ocena prekritosti regije nagnjena k višjim vrednostim in se šele v nekaj naslednjih slikah stanje umiri. Za realnejšo oceno zato prvih 10 slik ocenimo z najnižjo vrednostjo.
Kot izhodni rezultat dobimo opis tirnice poteka sledenja. Tirnico razdelimo na segmente, kjer ni prišlo do odpovedi sledilnika. Segmentom, ki so krajši od dolžine N_s se doda ničle, daljše segmente pa se poreže na dano dolžino. Vsi segmenti, ki so se končali brez odpovedi sledilnika in so krajši od N_s so odstranjeni. Izračun ponovimo za vse dolžine $N_s = N_{lo} : N_{hi}$ in s tem dobimo okvirno pričakovano povprečje prekrivanja regije $\hat{\Phi}$.

$$\hat{\Phi} = \frac{1}{N_{hi} - N_{lo}} \sum_{N_s = N_{lo}:N_{hi}} \hat{\Phi}_{N_s} \tag{3.1}$$

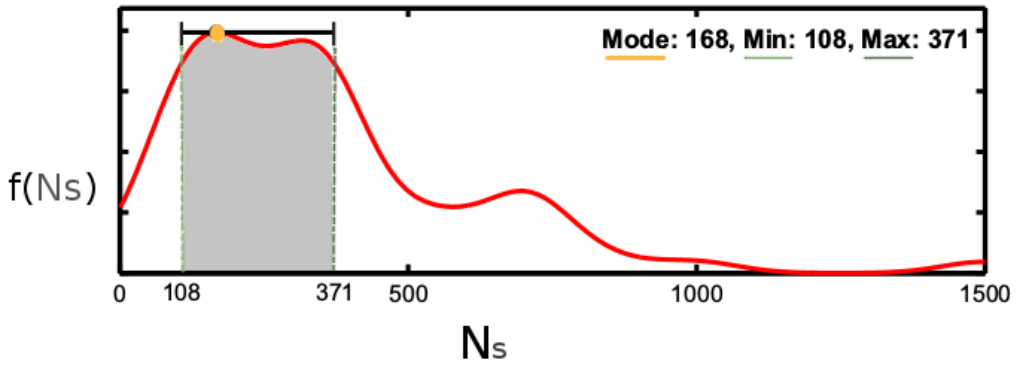
Slika 3.5 na levi stani prikazuje neobdelani izhodni rezultat sledenja na izbranih sekvencah. Rezultate segmentiramo na točkah odpovedi sledilnika in za vsak segment posebej izračunamo njegovo povprečno prekrivanje regije.

Razpon tipične dolžine sekvenc $[N_{lo}, N_{hi}]$ je bil pridobljen iz funkcije gostote porazdelitvene verjetnosti (ang., probability density function, PDF, PD function) nad dolžinami vseh sekvenc, izračunane s pomočjo ocene gostote jedra (ang., kernel density estimation, KDE) [23, 24]. Spodnja in zgornja meja sta bili določeni kot prvi točki levo in desno od modusa PD funk-



Slika 3.5: Segmentacija tirnic [25].

cije, ki zadovoljuje pogojema $p(N_{lo}) \approx p(N_{hi})$ in je $P[N_{lo} \leq X \leq N_{hi}] = \sum_{X=N_{lo}}^{N_{hi}} Xf(X) \approx 0.5$. Slika 3.6 prikazuje PD funkcijo dolžine sekvenc. Mo-



Slika 3.6: Spodnja in zgornja meja dolžine sekvenc [25].

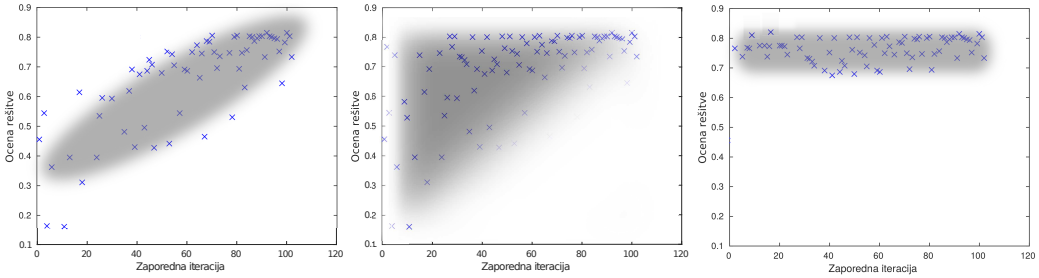
dus, ali maksimum funkcije PD je obarvan oranžno, z zeleno barvo pa sta prikazana minimum in maksimum razpona. S sivo je obarvana ploščina pod krivuljo od minimuma do maksimuma, ki predstavlja 5% celotne površine.

3.5 Grafični prikaz rezultatov

V tem podpoglavju so opisani predlagani pristopi vizualizacije rezultatov in njihova interpretacija.

3.5.1 Razpršenost rešitev

Prikaz razpršenosti rešitev omogoča pregled nad porazdelitvijo rezultatov skozi evaluirane kombinacije vrednosti parametrov. Predstavlja hiter pregled nad uspešnostjo optimizacijske metode in okvirno oceno ali je izboljšanje rezultatov z izbranim sledilnikom pravzaprav možna.



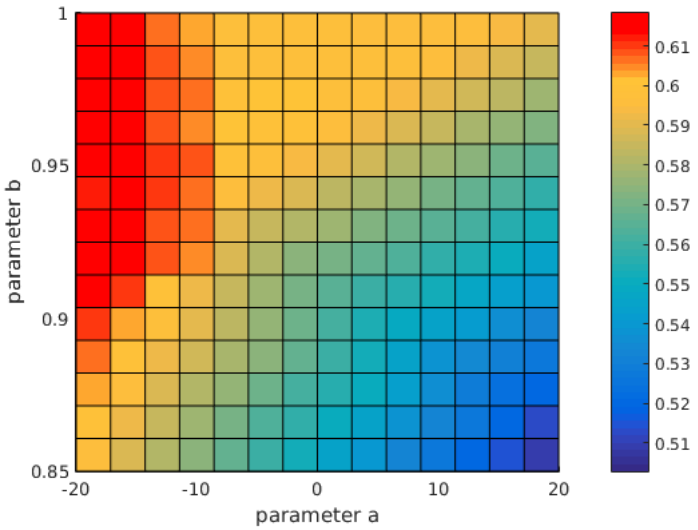
Slika 3.7: Zelo dobra (levo), dobra in slaba (desno) razpršenost.

V sliki 3.7 so prikazane tri različne porazdelitve rezultatov. Z modrimi križci so označeni podatki in s sivo je označeno območje rešitev. Iz prvega primera je razvidno, da glede na zaporedno iteracijo, ocena rešitve po večini narašča. Iz ocene začetnih vrednosti parametrov v prvi iteraciji, smo z optimizacijsko metodo rezultat občutno izboljšali. V drugem primeru do večjih sprememb ni prihajalo. V prvih nekaj iteracijah smo prišli zelo blizu optimalni rešitvi. Možnost je, da smo obtičali v lokalnem optimumu ali, pa z danim sledilnikom rezultata ne moremo izboljšati. V zadnjem primeru pa je zelo dobro razvidno, da izboljšave ni bilo. Najverjetneje, z nastavljanjem parametrov sledilnika, na izbrani sekvenci ne moremo zagotoviti uspešnejšega sledenja.

3.5.2 Odvisnosti med parametri

Grafična predstavitev odvisnosti med parametri je prikazana kot toplotni graf (ang., heat map, heatmap). Heatmap je dvodimenzionalna grafična predstavitev podatkov, kjer so vrednosti predstavlje kot barve. Izbor takšne

predstavitve je primeren za kompleksnejše skupine podatkov in prikaza odvisnosti med njimi, saj omogoča njihovo hitro in enostavno interpretacijo. Na lažje razumevanje lahko vplivamo tudi z izborom barvne lestvice v kateri so podatki prikazani.

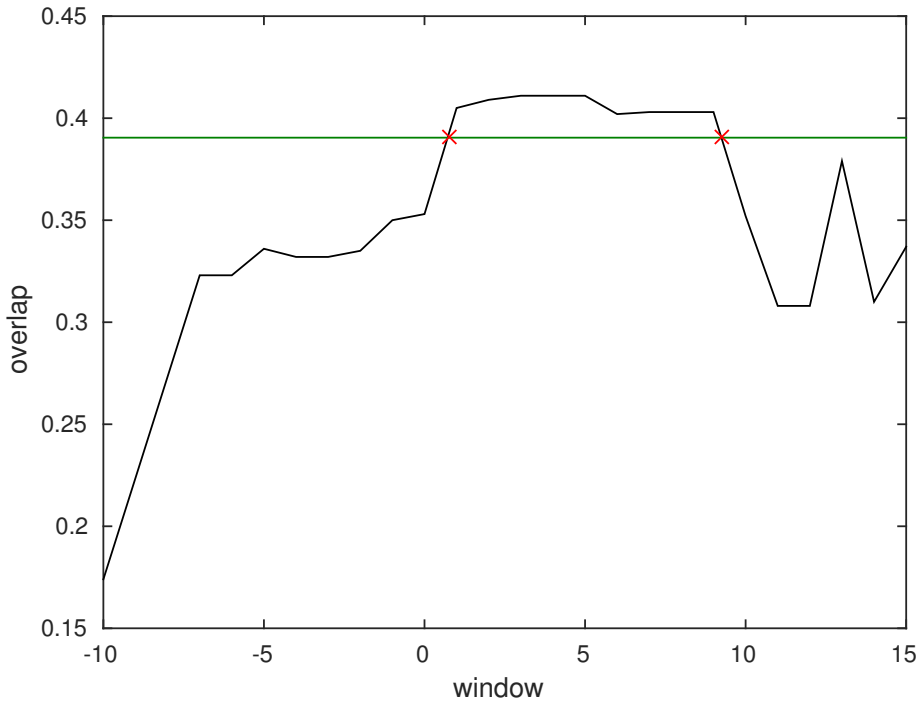


Slika 3.8: Primer toplotnega grafa.

Pri izboru barv moramo biti pozornosti, da izbor barvne lestvice prikaže, kar želimo izpostaviti. V našem primeru želimo izpostaviti območje vrednosti parametrov, kjer je bilo sledenje najbolje ocenjeno. Uporabljena je bila rdeča-rumena-modra lestvica. Je ena izmed najlažje predstavljivih lestvic, saj uporablja barvno interpretacijo temperatur. Za višje vrednosti so uporabljene toplejše barve in za nižje, hladnejše barve. Slika 3.8 prikazuje primer take predstavitve in je opremljena z barvno lestvico ter njenimi pripadajočimi vrednostmi. Zelo hitro lahko ugotovimo, kje je bila ocena sledenja največja. S tako predstavitvijo enostvno dobimo tudi občutek povezave med parametri in velikost območja, kjer je sledenje uspešno.

3.5.3 Stabilnost parametrov

Stabilnost posameznega parametra p_i testiramo tako, da spreminjamo njegove vrednosti, vrednosti ostalih parametrov pa fiksiramo. Za začetno oceno intervala uporabimo linearno interpolacijo nad vsemi podatki. Nato s korakom d na dobljenem intervalu iščemo presečišča med interpolirano funkcijo in mejo T_{min}, da zadostuje pogoju $overlap(x) \simeq T_{min}$. Mejo, za katero še smatramo, da je bilo sledenje uspešno postavimo na 5% maksimalne vrednosti funkcije.



Slika 3.9: Stabilnost parametra.

Slika 3.9 predstavlja primer takega grafa. Krivuja predstavlja linearno interpolacijo nad podatki. Meja T_{min} je določena z zeleno horizontalo in najdeni zgornja in spodnja meja sta prikazani z rdečim križcem. Pri takem prikazu, lahko zelo hitro določimo kako stabilen je parameter. Večja kot je razdalja med obema križcema, bolj je parameter stabilen. Vrednost

lahko znotraj intervala premikamo, brez prevelikih sprememb ocene sledenja. Nasprotno, manjša je razdalja, manj je parameter fleksibilen. Z majhno spremembo vrednosti lahko konkretno vplivamo na potek sledenja. Iz tega sledi, da je parameter nestabilen in posledično zelo pomemben.

Poglavje 4

Eksperimentalna evaluacija

V tem poglavju je opisan način testiranja sledilnika z uporabo nadgrajenega sistema TraXtor z našim novim podsistemom za analizo parametrov, opisi eksperimentov ter prikaz in analiza rezultatov. Testiranje je bilo opravljeno s sledilnikom Mean Shift z adaptacijo velikosti (ang., Adaptive-Scale Mean Shift, ASMS) [40], kateri je implementiran v programskem jeziku C++ [39] z uporabo knjižnice OpenCV [2] za omogočanje kar se da hitrega izvajanja algoritma. Testiranje sledilnika je bilo opravljeno na zbirki sekvenc VOT Challange 2013 [36]. Sledilnik ASMS v osnovi pri sledenju zelo uspešen in je bil izbran zaradi njegove enostavnosti in hitre izvedbe. Čeprav je ASMS neparametričen algoritem, lahko z izborom nekaj parametrov posredno vplivamo na uspešnost sledenja. Zaradi njegove hitre izvedbe in majhnega števila parametrov je tako zelo primeren za naloge testiranja. Analiza rezultatov je tako lažja za iterpretacijo, vendar je postopke možno aplicirati tudi na kompleksnejše sledilnike.

4.1 Testiranje

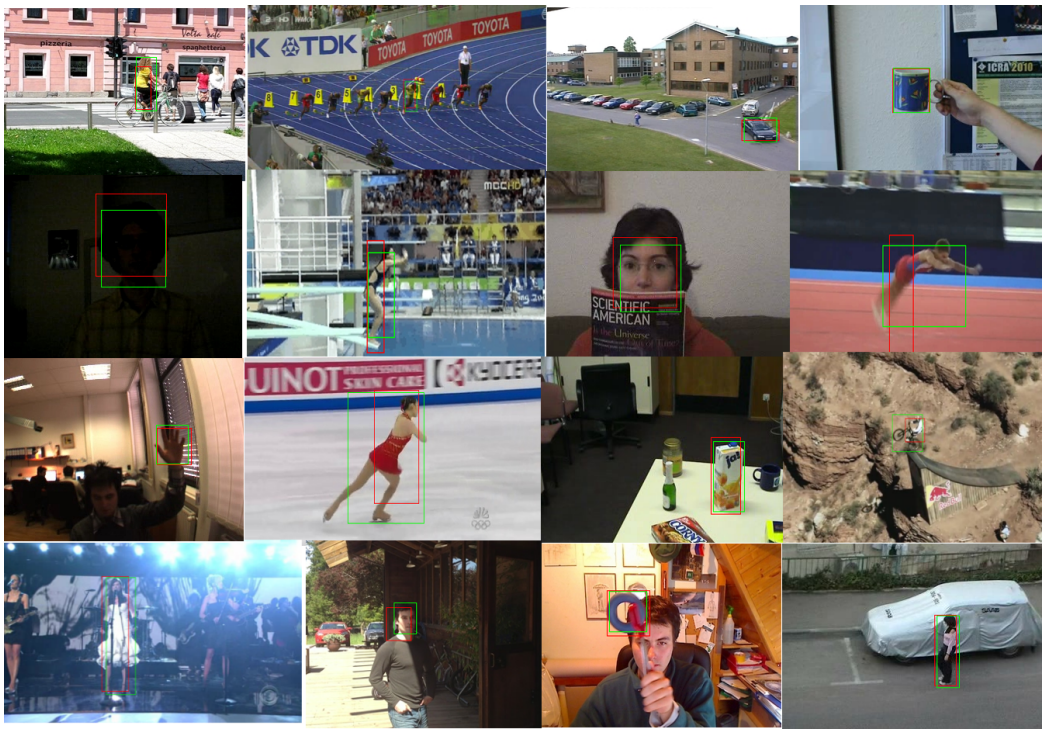
Iskanje optimalnega nabora vrednosti parametrov je bilo testirano z novo hibridno metodo ocenjevanja sledilnikov, predstavljeno v Poglavju 3, Podpoglavje 3.4 na javno dostopni zbirki sekvenc VOT2013 [36]. Sledilnik je bil testiran na vseh sekvencah hkrati, na posameznih sekvencah in na skupinah sekvenc glede na njihove lastnosti. V nadaljevanju so podrobneje opisane skupine sekvenc in kako so bile izbrane. V Podpoglavju 4.2 so predstavljeni rezultati v grafični in tabelni obliki ter analiza dobljenih rezultatov.

4.1.1 Izbrane sekvence

Zbirka sekvenc VOT2013 [36] je primerna, saj vsebuje le 16 sekvenc, katere so dovolj raznolike, tako glede na težavnost kot lastnosti sekvenc. Lastnosti sekvence predstavljajo atributi, kot so premikanje kamere, sprememba svetlobe, prekrivanje objekta, spremembe velikosti ali deformacija sledenega objekta.V Tabeli 4.1 so naštete vse sekvence zbirke in njihove težavnosti.

Sekvenca	Težavnost
Bicycle	lahka
Bolt	težka
Car	lahka
Cup	lahka
David	lahka
Diving	težka
Face	srednje lahka
Gymnastics	srednje težka
Hand	težka
Iceskater	srednje težka
Juice	srednje lahka
Jump	srednje lahka
Singer	srednje lahka
Sunshade	srednje lahka
Torus	srednje težka
Woman	srednje lahka

Tabela 4.1: Sekvence VOT2013 [36].



Slika 4.1: Zbirka sekvenc VOT2013 [36].

Slika 4.2: Sekvenci *Bolt* (zgoraj) in *Cup* (spodaj).

Slika 4.1 vsebuje po eno sliko vsake sekvence po vrsti iz Tabele 4.1 in si sledijo od leve proti desni, od zgoraj navzdol. Rezultati, kjer je sekvenca poimenovana *All*, pomeni da je bilo testiranje narejeno preko vseh sekvenc v zbirki. V Sliki 4.2 sta izpostavljeni dve sekvenci, težja *Bolt* in lažja *Cup*, ki sta bili v nadaljevanju uporabljeni.

4.2 Rezultati

4.2.1 Primerjava rezultatov

V Tabeli 4.2 so prikazani rezultati $\hat{\Phi}_{opt}$ pridobljeni s pomočjo optimizacijskega eksperimenta ter rezultati $\hat{\Phi}_{noOpt}$ brez optimizacije z uporabo privzetih vrednosti parametrov sledilnika ASMS (*window=0, dim=4, alpha=1*) [40]. Vrednost $\hat{\Phi}_{optParams}$ je bila izračunana z zagonom sledilnika nad posameznimi sekvencami z optimalnimi parametri iz vrstice *All*. Parametri *optParams* so bili pridobljeni z optimizacijo parametrov preko vseh sekvenc v zbirki.

Sekvenca	window	dim	alpha	$\hat{\Phi}_{opt}$	$\hat{\Phi}_{noOpt}$	$\hat{\Phi}_{optParams}$
Bicycle	7	4	0.99	**0.605**	0.568	0.191
Bolt	3	4	0.98	**0.408**	0.283	0.283
Car	0	4	0.99	**0.655**	**0.655**	0.607
Cup	-7	5	0.99	**0.816**	0.783	0.814
David	-10	6	0.99	**0.635**	0.272	0.434
Diving	0	5	0.99	**0.469**	**0.464**	0.462
Face	0	5	0.86	**0.736**	0.687	0.677
Gymnastics	3	5	1	**0.674**	0.654	0.632
Hand	0	4	1	**0.624**	**0.624**	0.601
Iceskater	7	3	1	**0.672**	0.592	0.569
Juice	7	4	1	**0.626**	0.625	0.582
Jump	3	2	1	**0.635**	0.488	0.397
Singer	-7	6	1	**0.624**	0.287	0.619
Sunshade	10	5	0.99	0.553	0.525	**0.560**
Torus	-3	3	1	**0.801**	0.781	0.749
Woman	-15	3	1	**0.650**	0.194	0.618
All	-7	5	1	0.365	0.317	0.549

Tabela 4.2: Primerjava rezultatov.

Z iskanjem optimialnih parametrov pri posamezni sekvenci, lahko v večini primerov pridemo do boljših rezultatov. To se pozna predvsem v težjih sekvencah kot so *Bolt, David in Woman*. Opaziti je, da metoda ocenjevanja sledilnika najbolj upošteva parametre, kjer so težje sekvence bolje ocenjene. Iz vrednosti parametrov lahko določimo tudi kar nekaj lastnosti sekvence. Opaženo je, da lahko v sekvencah, kjer je ozadje zelo razgibano, na uspešnost

sledenja vplivamo z zmanjšanjem sledilnega okna (Cup, David, Singer, Torus, Woman). V teh sekvencah se ozadje zelo spreminja in se ga z zmanjšanjem sledilnega okna izognemo. Največ sekvenc uporabi večje sledilno okno, kar potrjuje trditev, da sledilnik ASMS v splošnem deluje bolje z uporabo večjega okna. Pri sekvencah, kjer se objekt iz slike v sliko barvno zelo spreminja, lahko z manjšo vrednostjo alpha te spremembe upoštevamo in s tem izboljšamo rezultat sledenja. Kjer pa do konstantnih sprememb ne prihaja, nastavljanje vrednosti alpha nima velikega vpliva. Sekvence, kjer so barve objekta in ozadja jasno ločene, opazimo da je ocena sledenja boljša z manjšim številom košev histograma. Kjer pa je barvna razporeditev tako objekta kot ozadja podobna, moramo slediti z večjim številom košev, da zagotovimo bolj natančno ločevanje barv.

Sekvenca	Acc_{asms}	Acc_{opt}	Rob_{asms}	Rob_{opt}
Bicycle	**0.568**	0.191	0	2
Bolt	0.283	0.283	2	2
Car	**0.655**	0.607	1	1
Cup	0.783	**0.814**	0	0
David	0.272	**0.434**	2	**1**
Diving	**0.464**	0.462	0	0
Face	**0.687**	0.677	0	0
Gymnastics	**0.654**	0.632	0	0
Hand	**0.624**	0.601	0	0
Iceskater	**0.592**	0.569	0	0
Juice	**0.625**	0.582	0	0
Jump	**0.488**	0.397	0	0
Singer	0.287	**0.619**	2	**0**
Sunshade	0.525	**0.560**	0	0
Torus	**0.781**	0.749	0	0
Woman	0.194	**0.618**	4	**1**
All	0.317	**0.365**	11	**8**

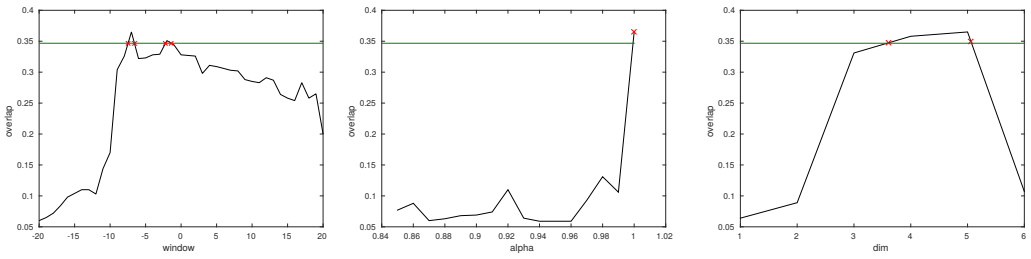
Tabela 4.3: Točnost in robustnost sledilnika.

Tabla 4.3 prikazuje rezultate točnosti (ang., accuracy, Acc) in robustnosti (ang., robustness, Rob) na primeru sledilnika ASMS. Polji Acc_{asms} in Rob_{asms} predstavljata zagon sledilika s predlaganimi vrednostmi [40] parametrov ($asms : [window = 0, dim = 4, alpha = 1]$). Polji Acc_{opt} in Rob_{opt} pa

z našimi parametri, pridobljenimi z optimizacijsko metodo ($opt : [window = -7, dim = 5, alpha = 1]$). Sivo obarvana polja predstvljajo sekvence, kjer je bila ocena sledenja z našimi parametri, boljša od ocene s predlaganimi parametri *asms*. Opaziti je, da so ocene z našimi parametri, po večini nekoliko slabše, vendar so primerjavi s paramteri *asms*, težje sekvence ocenjene veliko boljše. V primeru sekvence *Woman* je izboljšava še posebej očitna.

4.2.2 Stabilnost parametrov

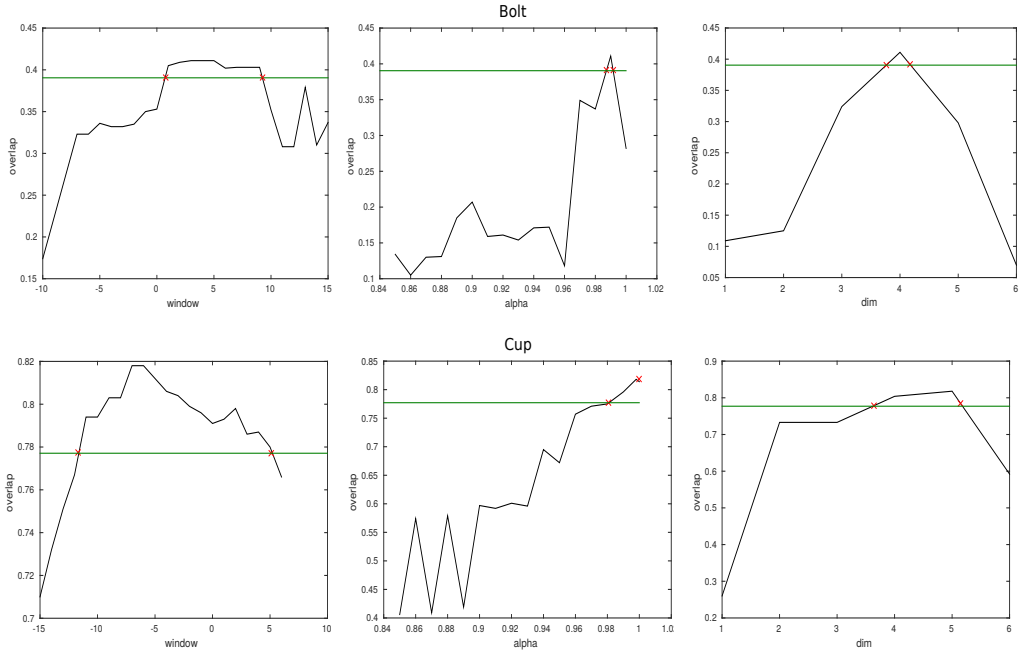
Stabilnost parametrov določimo tako, da pregledamo okolico trenutne vrednosti, kjer ostali parametri ostanejo enaki in jih ne spreminjamo. Če se za majhno spremembo vrednosti parametra rezultat občutno poslabša, pomeni, da je parameter nestabilen in že majhna sprememba povzroči slabše sledenje objekta. Dopustljiva meja T_{min}, za katero še smatramo, da je bilo sledenje uspešno smo nastavili na pet odstotkov maksimalne ocene sledilnika pri posameznem testiranju. Za izračun intervala dopustljivih vrednosti, je bila uporabljena linearna interpolacija, kjer iščemo $overlap(x) \simeq T_{min}$ in je x, vrednost parametra.Predstavljeni so rezultati testiranja na vseh sekvencah skupaj, na težji sekvenci *Bolt* in lažji *Cup*.



Slika 4.3: Testiranje na vseh sekvencah skupaj.

Pri testiranju na vseh sekvencah hkrati so razponi parametrov ozki (Slika 4.1), kar je razumljivo, saj morajo zadovoljiti lastnostim različnih sekvenc. Zaradi razgibanosti težavnosti sekvenc in njihove raznolikosti, so optimalni parametri za vsako sekvenco zelo različni. V primeru okna je vidno, da s spreminjanjem vrednosti rezultati bistveno ne spreminjajo hitro. Nasprotno je v

primeru izbora vrednosti *alpha* in spreminjanjem števila košev histograma.



Slika 4.4: Težja sekvenca (*Bolt*).

V primeru lažje sekvence je opazno, da je nabor dopustljivih vrednosti večji, v primerjavi z težjo sekvenco *Bolt*. V težji sekvenci je tako dovoljiva le sprememba velikosti okna, parametra alpha in dim sta določena z enolično rešitvijo. To pomeni, da sta parametra alpha in dim v primeru težje sekvence nestabilna in posledično zelo pomembna. V primeru lažje sekvence pa so vsi parametri določeni z večjim dopustnim naborom vrednosti. Primerjava rezultatov je pokazala, da je najstabilnejši parameter velikost okna. Ta dopušča največjo spremembo vrednosti in je zato najmanj pomemben parameter. Za najmanj stabilnega se je izkazal parameter alpha, ki ima nabor vrednosti zelo omejen. Izbor fiksnih parametrov v sledilniku ASMS je znotraj dovoljenih vrednosti.

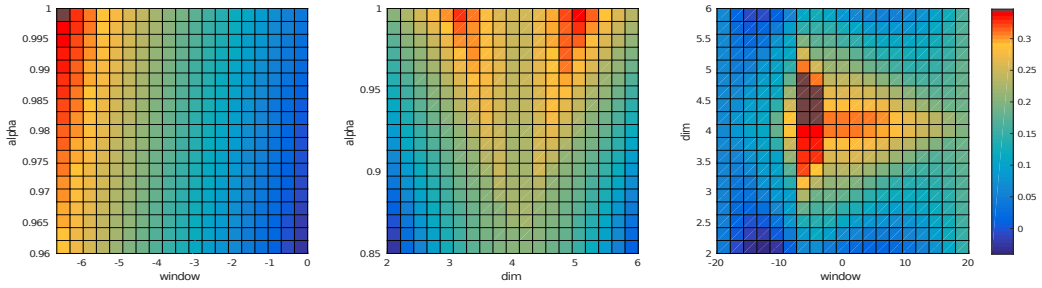
Sekvenca	Parameter	Razpon	*optParam*	ASMS
All	window	[-7.4675, -6.5776] in [-2.1912, -1.3994]	-7	0
	alpha	1	1	1
	dim	[⌈3.6061⌉, ⌊5⌋]	5	4
Bolt	window	[0.7608 - 9.2793]	3	0
	alpha	[0.9871, 0.9916]	0.988	1
	dim	[⌈3.767⌉, ⌊4.171⌋]	4	4
Cup	window	[-11.6892, 5.1216]	-7	0
	alpha	[0.9806, 1]	0.99	1
	dim	[⌈3.6364⌉, ⌊5.1515⌋]	5	4

Tabela 4.4: Razpon parametrov.

V Tabeli 4.4 so prikazani razponi vrednosti posameznih parametrov, pridobljeni s pomočjo interpolacije na neobdelanih podatkih. V polju $optParams$ so vrednosti parametrov, ki smo jih dobili z optimizacijo in v polju $ASMS$ so vrednosti, ki so predvidene za sledilnik ASMS [40]. Vrednosti obarvane rdeče, predstavljajo vrednosti izven dopustljive meje.

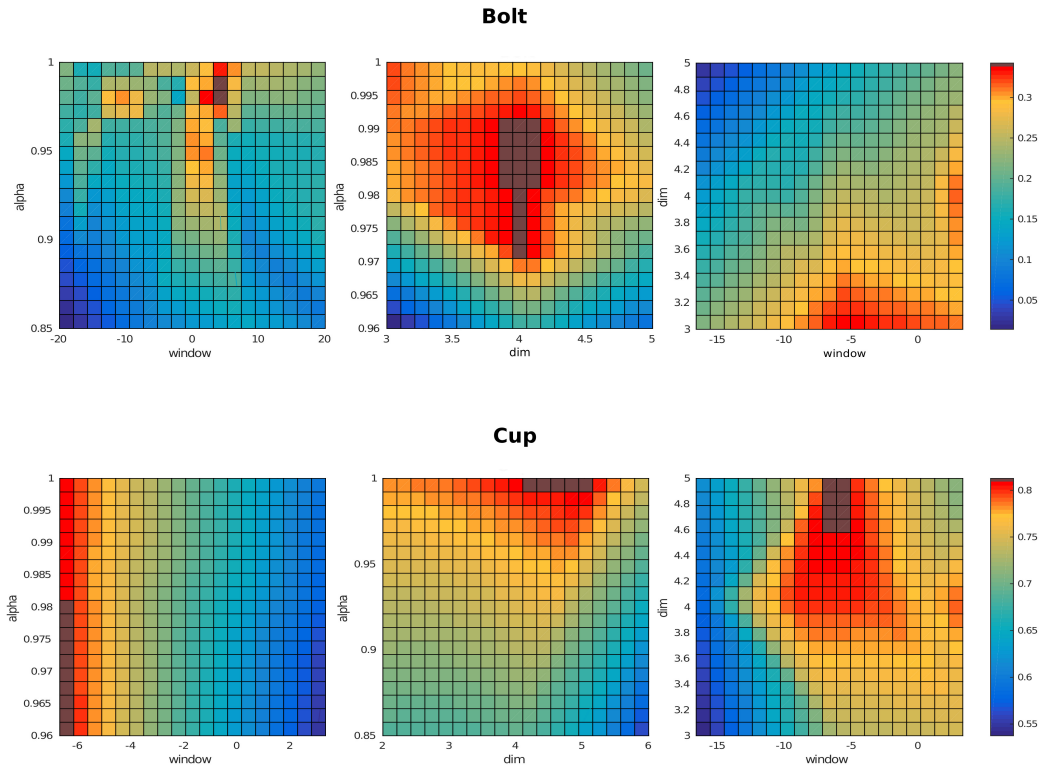
4.2.3 Odvisnosti parametrov

Odvisnosti med parametri so prikazane z uporabo toplotnih grafov (ang., heat map) in nam omogočajo hiter pregled nad tem, kako parametri vplivajo eden na drugega. Rjava območja prikazujejo najvišjo oceno sledenja in rdeča, območja kjer je bilo sledenje dobro ocenjeno. Nižje ocene so prikazane s hladnejšimi barvami. S tako predstavitvijo lahko hitro opazimo kako s spreminanjem vrednosti parametra vplivamo na oceno sledenja. Za vsako testiranje je ob grafih tudi barvna lestvica z njihovimi vrednostmi, ki omogočajo realnejši pogled nad barvno prestavitvijo. Odvisnosti so prikazanje nad pari parametrov, kjer je tretji parameter fiksiran na optimalno vrednost, pridobljeno z optimizacijsko metodo (Tabela 4.2).



Slika 4.5: Testiranje na vseh sekvencah skupaj.

Pri testiranju na vseh sekvencah skupaj je najbolj zanimiv vpliv števila košev na vrednost *alpha*. Opazimo, da obstajata dva maksimuma. In sicer pri številu košev 2^3 in 2^5 z večjo vrednostjo *alpha*. Zanimivo je, da moramo z večjim številom košev histograma model bolj modificirati, z manjšim številom košev, pa modela ne smemo veliko spreminjati. Iz grafov je razvidno, da je rezultat zelo odvisen od velikosti okna, kateri je omejen na zelo majhen interval. To lahko opazimo v prvem in tretjem grafu, kjer z majhno spremembo velikosti okna hitro pademo izven območja dobre rešitve. V primeru, da je parameter *window* v območju visoke vrednosti, pa z ostalimi parametri nismo tako omejeni, in jih njihov razpon razmeroma velik.

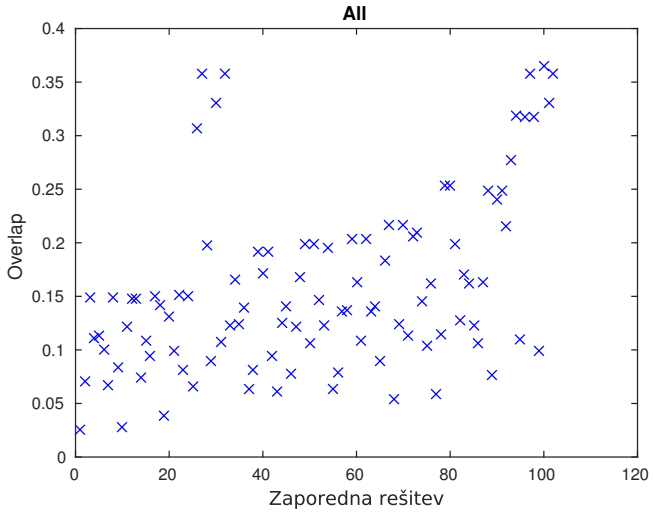


Slika 4.6: Težja sekvenca *Bolt* (zgoraj) in lažja sekvenca *Cup* (spodaj).

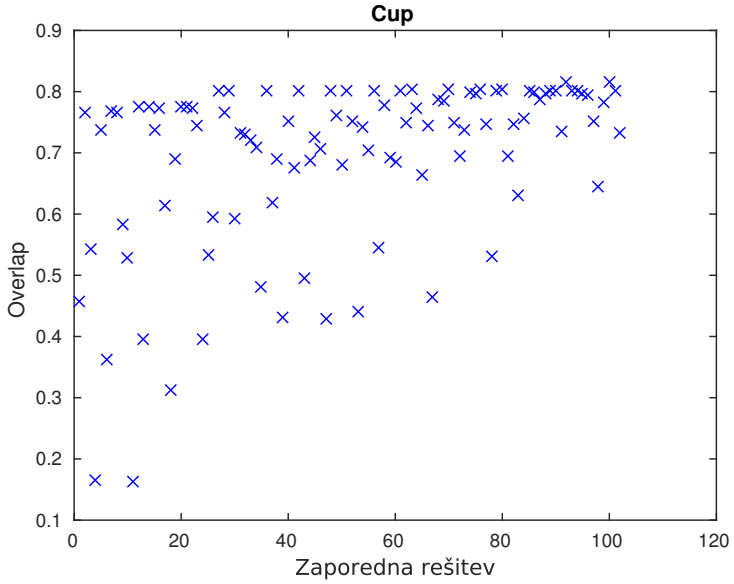
Slika 4.6 vsebuje 6 grafov. Zgornji trije predstavljajo odvisnosti med parametri na težji sekvenci *Bolt* in spodnji trije, odvisnosti med parametri na lažji sekvenci *Cup*. V primeru težje sekvence lahko opazimo, da je odvisnost med parametrom *alpha* in *window* zelo močna. Imamo majhno območje, kjer je bilo sledenje uspešno. S spreminjanjem katerega koli izmed teh dveh parametrov konkretno zniža oceno uspešnosti. S primerjavo rezultatov opazimo, da so v težjih sekvencah območja dobrega sledenja bolj omejena in so odvisnosti parametrov večje. Pri sekvencah z manj zahtevnimi lastnostmi pa so ta območja večja.

4.2.4 Razpršenost rezultatov

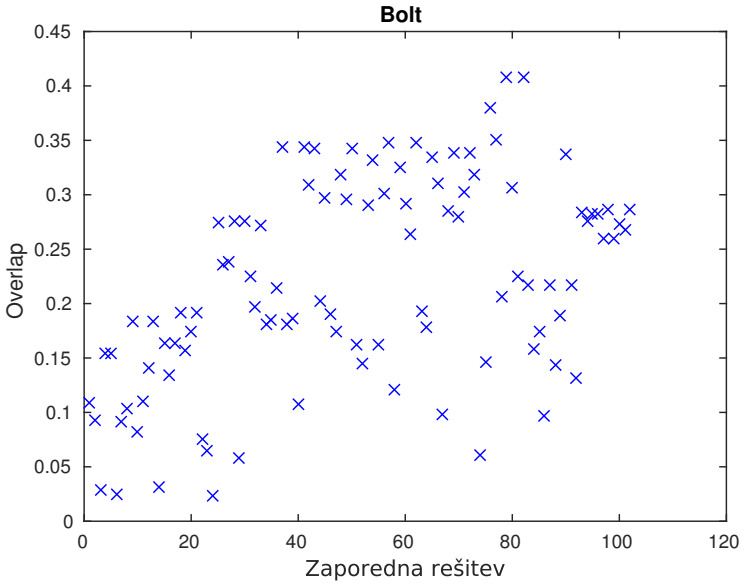
Za vizualizacijo uspešnosti metode je predlagan pristop razpršenosti rezultatov. Grafi prikazujejo potek iskanja optimalne rešitve in nam podajo informacijo za koliko so se rešitve glede na iteracijo izboljšale. Pri testiranju nad vsemi sekvencami je izboljšava rezultata očitna. Na Sliki 4.7 vidimo, da so najboljše rešitve zelo osamele, iz česar lahko slepamo, da je nastavitev parametrov za dan nabor sekvenc težaven. Sklepamo lahko tudi, da bo interval dovoljenih vrednosti ozek in bo rešitev zelo nestabilna. Iz Table 4.4 je razvidno, da je sklep utemeljen. Enako lahko opazimo pri težji sekvenci *Bolt*. Na primeru te sekvence lahko na Sliki 4.9 Opazimo linearno naraščanje ocene sledenja glede na zaporedno iteracijo, vendar z velikimi preskoki v oceni iz ene v drugo rešitev. Tudi iz tega lahko sklepamo, da je je rešitev ozko določena in bodo parametri razmeroma nestablni. V nasprotju, pa lahko na primeru lažje sekvence *Cup* na Sliki 4.8 opazimo, da so rešitve razporejene zelo enakomerno po liniji najvišje ocene. V začetnih iteracijah ocene rešitve sicer malo nihajo, vednar se ob koncu optimizacije stabilizirajo. Iz tega prikaza lahko sklepamo, da je rešitev na dani sekvenci le ena izmed mnogih, kar pomeni, da bodo parametri manj občutljivi na spremembe njhihovih vrednosti.



Slika 4.7: Testiranje na vseh sekvencah skupaj.



Slika 4.8: Lažja sekvenca (*Cup*).



Slika 4.9: Težja sekvenca (*Bolt*).

4.2.5 Povzetek

Glede na rezultate je ugotovljeno, da smo z optimizacijo parametrov sledilnika, uspešno izboljšali potek sledenja. Pri težjih sekvencah lahko na rezultat občutno vplivamo s skrbno izbranimi vrednostmi parametrov. V nekaterih primerih je prišlo tudi do več kot 200% izboljšave (na sekvenci $Woman$), vendar je to le rezultat, kjer smo parametre določili na podlagi ene same sekvence. Ocene so bile pridobljene na podlagi predlagane mere pričakovanega povprečnega preseka regij. V Tabeli 4.5 lahko vidimo, da je možno, z uporabo optimizacije parametrov sledilnika, doseči boljše rezultate. $ASMS_{default}$ pomeni zagon sledilnika s privzetimi parametri, ki jih je določil avtor [40]. $ASMS_{opt}$, pa zagon sledilnika z našimi optimiziranimi parametri. Za določitev izboljšave sledenja z optimizacijo parametrov, sta predlagani dve oceni. In sicer, povprečje preko vseh sekvenc $\overline{\Phi_{seq}}$ in ocena pričakovanega povprečnega preseka regij Φ_{all} nad vsemi sekvencami. Izboljšava z uporabo optimizacije je izražena v odstotkih.

	$ASMS_{default}$	$ASMS_{opt}$	Izboljšava (%)
$\overline{\Phi_{seq}}$	0.53	0.549	3.5
Φ_{all}	0.317	0.365	15

Tabela 4.5: Izboljšava rezultatov.

Poglavje 5

Sklepne ugotovitve

V diplomski nalogi smo se ukvarji s problemom določanja parametrov sledilnika, kjer je točnost in robustnost sledenja najboljša. Ker avtorji sledilnikov parametre nastavijo ročno, se postavi vprašanje ali so izbrani parametri zares najboljši. Pojavi se tudi vprašanje ali lahko določimo postopek s katerim lahko pridemo do optimalne rešitve.

Cilj diplomske naloge je bil razširiti pogled na pomembnost izbora parametrov in pomen, ki ga imajo pri sledenju. Postavilo se je tudi vprašanje, kaj izbrani parametri lahko povedo o lastnostih sekvenc in objektu, ki ga sledimo. Problem določanja optimalnih vrednosti smo definirali kot nalogo diskretne optimizacije in se reševanja lotili z uporabo preiskovalne metode najstrmejšega vzpona (ang., Steepest-Ascent Hill-Climbing, SAHC). Preizkusimo novo hibridno mero ocenjevanja sledilnikov, ki predstavlja pričakovano povprečno prekrivanje regije (ang., expected average overlap, EAO). Za analizo parametrov so predlagani postopki določanja stabilnosti parametrov, njihovih lastnosti in odvisnosti med parametri. Grafično prikažemo tudi uspešnost izbrane preiskovalne metode.

Ugotovljeno je bilo, da izbrana optimizacijska metoda v večini primerov, na posamezni sekvenci najde boljšo rešitev. Kjer rešitve ni bilo moč izboljšati, obstaja velika verjetnost, da izbrani sledili algoritem ni primeren za sledenje na dani sekvenci. Možno je tudi, da je metoda zgrešila globalni maksimum

in se ustavila v lokalnem. Največje izboljšave, je opaziti pri težjih sekvencah, kot je *Woman*, kjer je prišlo do več kot 200% izboljšave ocene sledenja. S pomočjo analize parametrov in grafične predstavitve, smo lahko zelo dobro opisali pomen parametrov. Pokazalo se je, da parametri vsebujejo ogromno informacij o lastnostih sekvenc in sledenih objektov. Zanimalo nas je, ali lahko določimo optimalne vrednosti parametrov, ki bi v splošem izboljšale potek sledenja in za koliko se razlikujejo od predlaganih. Z optimizacijsko metodo smo, na podlagi predlagane mere ocenjevanja sledilnikov, uspeli poiskati vrednosti parametrov, ki so v povprečju, uspešnost sledenja izboljšali za 15%.

5.1 Nadaljnje delo

Možnosti za nadgradnjo je veliko, saj je področje široko in zelo malo obravnavano. Metodo bi bilo zanimivo testirati na parametričnih sledilnikih, kjer je parametrov več in jih je velikokrat težje interpretirati. Trenutno je vizualizacija rezultatov ločena od sistema TraXtor, v katerem je implementiran naš pristop, zato je možnost nadgradje, vključitev samodejne analize podatkov.

Slike

Tabele

Literatura

[1] Vladimir Batagelj. Optimizacijske metode, 1997.

[2] G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Luka Čehovin. Anotator sekvenc Aibu. `https://github.com/votchallenge/aibu`. [Dostopano: 5.9.2016].

[4] Luka Čehovin. Knjižnica Coffeshop. Dosegljivo: `https://github.com/lukacu/coffeeshop`. [Dostopano: 5.9.2016].

[5] Luka Čehovin. Odprtokodna knjižnica za integracijo protokola TraX z obstoječimi sledilniki. Dosegljivo: `https://github.com/votchallenge/trax`. [Dostopano: 5.9.2016].

[6] Luka Čehovin. Visual object tracking toolkit documentation. Dosegljivo: `http://box.vicos.si/vot/toolkit/docs/`. [Dostopano: 5.9.2016].

[7] Luka Čehovin. *A Hierarchical Adaptive Model for Robust Short-term Visual Tracking: A Dissertation*. PhD thesis, L. Čehovin, 2015.

[8] Luka Cehovin, Matej Kristan, and Ale Leonardis. Is my new tracker really better than yours? In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 540–547. IEEE, 2014.

[9] Luka Čehovin, Matej Kristan, and Aleš Leonardis. An adaptive coupled-layer visual model for robust visual tracking. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1363–1370. IEEE, 2011.

[10] Luka Čehovin, Matej Kristan, and Aleš Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(4):941–953, 2013.

[11] Luka Čehovin, Aleš Leonardis, and Matej Kristan. Visual object tracking performance measures revisited. *arXiv preprint arXiv:1502.05803*, 2015.

[12] Luka Čehovin and Tomas Vojir. Visual object tracking toolkit. Dosegljivo: `https://github.com/votchallenge/vot-toolkit`. [Dostopano: 5.9.2016].

[13] Duc Phu Chau, Julien Badie, François Brémond, and Monique Thonnat. Online tracking parameter adaptation based on evaluation. In *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*, pages 189–194. IEEE, 2013.

[14] Duc Phu Chau, Monique Thonnat, and François Bremond. Automatic parameter adaptation for multi-object tracking. In *Computer Vision Systems*, pages 244–253. Springer, 2013.

[15] Yizong Cheng. Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790–799, 1995.

[16] Robert T Collins. Mean-shift blob tracking through scale space. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–234. IEEE, 2003.

[17] Dorin Comaniciu and Peter Meer. Mean shift analysis and applications. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1197–1203. IEEE, 1999.

[18] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577, 2003.

[19] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[20] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.

[21] Bahadir Karasulu and Serdar Korukoglu. A software for performance evaluation and comparison of people detection and tracking methods in video processing. *Multimedia Tools and Applications*, 55(3):677–723, 2011.

[22] Matej Kristan, Stanislav Kovačič, Ale Leonardis, and Janez Perš. A two-stage dynamic model for visual tracking. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 40(6):1505–1520, 2010.

[23] Matej Kristan, Aleš Leonardis, and Danijel Skočaj. Multivariate online kernel density estimation with gaussian kernels. *Pattern Recognition*, 44(10):2630–2642, 2011.

[24] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–23, 2015.

[25] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015: Challange and results.

Dosegljivo: http://www.votchallenge.net/vot2015/download/vot_2015_presentation.pdf. [Dostopano: 5.9.2016].

[26] Matej Kristan, Jiri Matas, Ales Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Cehovin. A novel performance evaluation methodology for single-target trackers. *arXiv preprint arXiv:1503.01313*, 2015.

[27] Matej Kristan, Janez Perš, Matej Perše, and Stanislav Kovačič. Closed-world tracking of multiple interacting targets for indoor-sports applications. *Computer Vision and Image Understanding*, 113(5):598–611, 2009.

[28] Junseok Kwon and Kyoung Mu Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1208–1215. IEEE, 2009.

[29] A Li, M Lin, Y Wu, MH Yang, and S Yan. NUS-PRO: A New Visual Tracking Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):335–349, 2016.

[30] Annan Li, Min Lin, Yi Wu, Ming-Hsuan Yang, and Shuicheng Yan. Nus-pro: A new visual tracking challenge. 2015.

[31] Hanxi Li, Chunhua Shen, and Qinfeng Shi. Real-time visual tracking using compressive sensing. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1305–1312. IEEE, 2011.

[32] Tahir Nawaz and Andrea Cavallaro. Pft: A protocol for evaluating video trackers. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 2325–2328. IEEE, 2011.

[33] Tasin Nawaz and Andrea Cavallaro. A protocol for evaluating video trackers under real-world conditions. *Image Processing, IEEE Transactions on*, 22(4):1354–1361, 2013.

[34] Ning Song Peng, Jie Yang, and Zhi Liu. Mean shift blob tracking with kernel histogram filtering and hypothesis testing. *Pattern Recognition Letters*, 26(5):605–614, 2005.

[35] Roman Pflugfelder, Matej Kristan, Aleš Leonardis, and Jiri Matas. Visual Object Tracking challange. Dosegljivo: `http://www.votchallange.net`, 2015. [Dostopano: 5.9.2016].

[36] Roman Pflugfelder, Matej Kristan, Aleš Leonardis, Jiri Matas, and Fatih Porikli. Visual Object Tracking challange. Dosegljivo: `http://www.votchallange.net/vot2013`, 2013. [Dostopano: 5.9.2016].

[37] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

[38] Andrew Tuson. Optimisation with hillclimbing on steroids: an overview of neighbourhood search techniques. *DAI RESEARCH PAPER*, 22, 1998.

[39] Tomas Vojir, Jana Noskova, and Jiri Matas. Robust scale-adpative mean-shift for tracking. Dosegljivo: `https://github.com/vojirt/asms`. [Dostopano: 5.9.2016].

[40] Tomas Vojir, Jana Noskova, and Jiri Matas. Robust scale-adaptive mean-shift for tracking. *Pattern Recognition Letters*, 49:250–258, 2014.

[41] Naiyan Wang, Jianping Shi, Dit-Yan Yeung, and Jiaya Jia. Understanding and diagnosing visual tracking systems. *arXiv preprint arXiv:1504.06055*, 2015.

[42] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[43] Alper Yilmaz. Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007.

[44] Jie Zhao, Wen Qiao, and Guo-Zun Men. An approach based on mean shift and kalman filter for target tracking under occlusion. In *2009 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2058–2062. IEEE, 2009.

[45] Luka Čehovin. *TraX: Visual Tracking eXchange Protocol*, 2014.